世界著名计算机教材精选

PEARSON Prentice Hall

经典密码学与现代密码学

（美）Richard Spillman 著
叶阮健 曹 英 张长富 译



CLASSICAL AND CONTEMPORARY CRYPTOLOGY

清华大学出版社

世界著名计算机教材精选

经典密码学与现代密码学

(美)Richard Spillman 著
叶阮健 曹 英 张长富 译

清华大学出版社
北 京

Simplified Chinese edition copyright © 2005 by PEARSON EDUCATION ASIA LIMITED and TSINGHUA UNIVERSITY PRESS.

Original English language title from Proprietor's edition of the Work.

Original English language title: Classical and Contemporary Cryptology by Richard Spillman, Copyright © 2005

EISBN: 0-13-182831-2

All Rights Reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Prentice Hall PTR.

This edition is authorized for sale only in the People's Republic of China (excluding the Special Administrative Region of Hong Kong and Macao).

本书中文简体翻译版由 Pearson Education(培生教育出版集团)授权给清华大学出版社在中国境内(不包括中国香港、澳门特别行政区)出版发行。

北京市版权局著作权合同登记号 图字: 01-2004-5631

版权所有,翻印必究。举报电话: 010-62782989 13501256678 13801310933

本书封面贴有 Pearson Education(培生教育出版集团)激光防伪标签,无标签者不得销售。

图书在版编目(CIP)数据

经典密码学与现代密码学/(美)斯皮尔曼(Spillman, R.)著; 叶阮健,曹英,张长富译. · 北京: 清华大学出版社,2005. 7

(世界著名计算机教材精选)

书名原文: Classical and Contemporary Cryptology

ISBN 7-302-10740-8

Ⅰ. 经… Ⅱ. ①斯… ②叶… ③曹… ④张… Ⅲ. 密码—理论—高等学校—教材 Ⅳ. TN918.1

中国版本图书馆 CIP 数据核字(2005)第 027345 号

出 版 者: 清华大学出版社　　地　址: 北京清华大学学研大厦

http://www.tup.com.cn　　邮　编: 100084

社 总 机: 010-62770175　　客户服务: 010-62776969

印 刷 者: 北京市清华园胶印厂

装 订 者: 三河市化甲屯小学装订二厂

发 行 者: 新华书店总店北京发行所

开　本: 185×260　印张: 16.75　字数: 412 千字

版　次: 2005 年 7 月第 1 版　2005 年 7 月第 1 次印刷

书　号: ISBN 7-302-10740-8/TP · 7152

印　数: 1～3000

定　价: 35.00 元

前　　言

本书的目标是介绍密码学这一令人向往的世界。这是一个多面的世界，对一些人来说，它是一个侦探与保密的世界，对另一些人来说，它是数学与计算机的世界。无论你如何看待它，密码学都是神秘而富有冒险色彩的。它还超越了传统的学术学科。它不只是计算机科学的内容，密码学的研究包括历史、政治学、工程、语言军事学、伦理、数学和工业技术学等。任何单本书都不可能从所有这些方面来介绍密码学，因此，密码学的学生必须具有宽广的知识背景。本书只是作为学习这些知识和理解这些复杂但有益的内容的一个起点。

有两个原则指导本书的写作。第一个是密码学并不是始于计算机的发明。而现代密码学全部是基于计算机的，但它们很大程度上还是要归功于经典加密技术开发者的早期工作。这些开发者利用手工和铅笔工作，以发现经典加密法中的弱点。没有计算机，甚至没有计算器的帮助，他们必须训练成能认识加密的替换模式，进行数据组织。因此，要学习如何像译解密码者那样“思考”，你需要理解和意识到经典加密法的智慧和毅力。

第二个指导原则是，密码学课程不是(也不应该是)程序设计课程。编写一两个实现加密法或分析工具的程序可能对学生很有帮助，学习为所有重要的加密法和工具编写和调试代码将花费不少时间，这势必要减少真正学习密码学知识的时间。编写加密程序应该是算法或程序设计课程的任务。因此，与本书配套有一个软件包，即密码分析软件(CAP)，可以从 www. tup. com. cn 下载，它提供了经典加密法和现代加密法的实现，并含有一套分析这些加密法的工具。本教材和该软件一起将为你的密码学学习提供真实的动手体验。

密码学的初级学生、密码学爱好者和高级学生，都将从本书和配套的软件程序 CAP 中获益。第一部分介绍了密码学的经典问题，这对不熟悉密码学领域但准备开始学习的人很有帮助。高级学生可能会快速浏览一下如何运行 CAP 软件的内容，而将更多的时间花在那些不太熟悉的加密法和分析技术上。第二部分介绍了现代密码学，包括流加密法、块加密法和公共密钥加密法。这些内容可能对高级学生更有用。第三部分介绍了密码学的未来，并对量子加密法进行了简单介绍。

目　录

第1章　密码学概论……1
1.0　概述……1
1.1　密码学……1
1.2　重要术语……3
1.3　加密法的评价……3
1.4　密码分析法……4
1.5　编码与加密法的历史简介……5
1.6　经典加密法与现代加密法……7
1.7　CAP 软件介绍……7
1.8　本章小结……9
1.9　重要术语……10
习题……10

第一部分　经典加密法

第2章　经典单码加密法……15
2.0　概述……15
2.1　关键词加密法……15
2.1.1　关键词加密法的分析法……17
2.1.2　频率信息……17
2.1.3　使用 CAP 软件破解多关键词加密法……20
2.2　仿射加密法……24
2.2.1　仿射加密法的加密分析……24
2.3　多文字加密法……25
2.3.1　多文字加密法的分析……26
2.4　单码加密法的历史简介……27
2.5　本章小结……28
2.6　重要术语……29
习题……29
第3章　经典多码加密法……32
3.0　概述……32
3.1　Vigenere 加密法……32
3.1.1　Vigenere 加密法分析……34
3.1.2　用 CAP 分析 Vigenere 加密法……37
3.2　自动密钥加密法……40

3.2.1　自动密钥加密法的分析 …… 41
3.3　Nihilist 加密法 …… 43
3.4　圆柱面加密法 …… 44
3.4.1　Bazeries 圆柱面加密法的分析 …… 45
3.5　回转轮加密法 …… 48
3.5.1　Enigma 加密法的破解 …… 50
3.5.2　使用 CAP 软件破解回转轮加密法 …… 51
3.6　加密机的历史简介 …… 52
3.7　本章小结 …… 54
3.8　重要术语 …… 55
习题 …… 56

第 4 章　经典多图加密法 …… 58
4.0　概述 …… 58
4.1　Playfair 加密法 …… 58
4.1.1　Playfair 加密法分析 …… 60
4.1.2　用 CAP 软件分析 Playfair 加密法 …… 64
4.2　Hill 加密法 …… 65
4.2.1　在 CAP 软件中实现 Hill 加密法 …… 66
4.2.2　Hill 加密法分析 …… 66
4.3　Beale 加密法的历史简介 …… 68
4.4　本章小结 …… 71
4.5　重要术语 …… 72
习题 …… 72

第 5 章　经典换位加密法 …… 74
5.0　概述 …… 74
5.1　置换加密法 …… 75
5.1.1　置换加密法分析 …… 76
5.2　列置换加密法 …… 77
5.2.1　列换位加密法分析 …… 78
5.2.2　使用 CAP 软件来破解列换位加密法 …… 82
5.3　双重换位加密法 …… 83
5.3.1　双重换位加密法分析 …… 84
5.3.2　使用 CAP 软件破解双重换位加密法 …… 85
5.4　换位加密法的历史简介 …… 85
5.5　本章小结 …… 87
5.6　重要术语 …… 88
习题 …… 88

第二部分　现代加密法

第 6 章　流加密法 …… 93
6.0　概述 …… 93
6.0.1　计算机的特征 …… 93
6.0.2　XOR 逻辑运算 …… 94
6.1　流加密法 …… 95
6.1.1　线性反馈移位寄存器 …… 96
6.1.2　LFSR 周期分析 …… 97
6.1.3　随机位测试 …… 99
6.1.4　在 CAP 软件中实现流加密法 …… 100
6.2　破解流加密法 …… 101
6.2.1　用插入攻击法破解流加密法 …… 101
6.2.2　可能词攻击法一：位串匹配攻击法 …… 102
6.2.3　可能词攻击法二：单词匹配攻击法 …… 104
6.2.4　使用 CAP 软件破解流加密法 …… 105
6.3　其他流加密法的实现 …… 105
6.3.1　RC4 …… 106
6.3.2　评估 RC4 …… 109
6.3.3　A5 …… 109
6.3.4　单元自动操作 …… 110
6.3.5　生成随机数的其他方法 …… 114
6.4　一种不可破解的加密法 …… 114
6.5　实际应用 …… 114
6.6　流加密法历史简介 …… 115
6.7　本章小结 …… 116
6.8　重要术语 …… 117
习题 …… 117

第 7 章　块加密法 …… 119
7.0　概述 …… 119
7.1　块加密法的模式 …… 120
7.1.1　现代块加密法的模式 …… 121
7.1.2　关于填充问题 …… 122
7.2　乘积加密法 …… 123
7.2.1　块加密法的评估 …… 124
7.3　数据加密标准 …… 124
7.3.1　DES 密钥 …… 125
7.3.2　DES 的各个阶段 …… 126
7.3.3　DES 的其他实现 …… 131

7.3.4 CAP 软件的 DES 实现 …… 131

7.3.5 DES 分析 …… 133

7.4 IDEA …… 143

7.5 高级加密标准 …… 144

7.5.1 Rijndael 结构 …… 145

7.5.2 密钥生成 …… 150

7.5.3 AES 的操作 …… 151

7.5.4 AES 的安全性 …… 151

7.5.5 硬件实现 …… 152

7.5.6 其他候选加密算法 …… 154

7.6 块加密法的使用 …… 158

7.6.1 网络连接概述 …… 159

7.6.2 IPSec …… 160

7.7 块加密法的历史简介 …… 161

7.8 本章小结 …… 162

7.9 重要术语 …… 163

习题 …… 163

第 8 章 公钥加密法 …… 166

8.0 概述 …… 166

8.1 公钥的加密解密过程 …… 167

8.2 RSA …… 168

8.3 数字理论概述 …… 171

8.3.1 反模运算 …… 171

8.3.2 素数问题 …… 172

8.3.3 快速指数计算 …… 175

8.4 CAP 软件的 RSA 实现 …… 176

8.5 RSA 分析 …… 177

8.5.1 大整数的因子分解 …… 178

8.5.2 其他的 RSA 攻击法 …… 180

8.6 ElGamal 公钥系统 …… 183

8.6.1 生成器数字 …… 184

8.6.2 用 CAP 软件来实现 ElGamal …… 185

8.6.3 小结 …… 186

8.7 背包加密法 …… 186

8.7.1 破解背包加密法 …… 188

8.7.2 CAP 软件中的背包加密法 …… 190

8.8 椭圆曲线加密法 …… 191

8.8.1 算法评述 …… 193

8.9 公钥加密法的应用 …… 194

8.10　公钥加密法的历史简介 …… 195
8.11　本章小结 …… 195
8.12　重要术语 …… 196
习题 …… 196

第 9 章　密钥管理、数字签名、散列函数与证书 …… 200
9.0　概述 …… 200
9.1　密钥交换 …… 201
9.1.1　Internet 密钥交换过程 …… 203
9.1.2　组密钥 …… 204
9.1.3　广播加密 …… 208
9.2　可靠性 …… 210
9.3　数字签名 …… 212
9.3.1　散列函数 …… 213
9.3.1.1　MD5 …… 214
9.3.1.2　安全散列算法(SHA) …… 216
9.3.1.3　MD5 与 SHA-1 的比较 …… 218
9.3.1.4　基于块加密法的散列函数 …… 218
9.3.1.5　对散列函数的攻击 …… 219
9.3.1.6　CAP 中的散列函数 …… 221
9.3.2　盲签名 …… 222
9.3.3　数字签名标准 …… 225
9.4　公钥基础设施和证书 …… 227
9.4.1　建立证书 …… 228
9.4.2　证书内容 …… 229
9.4.3　使用证书 …… 230
9.4.4　证书的回收 …… 231
9.5　应用 …… 231
9.5.1　智能卡 …… 231
9.5.2　安全套接层 …… 232
9.6　历史回顾 …… 234
9.7　本章小结 …… 235
9.8　重要术语 …… 236
习题 …… 236

第三部分　密码学的未来

第 10 章　量子密码学 …… 243
10.0　概述 …… 243
10.1　量子系统概述 …… 243
10.1.1　量子位 …… 243

10.1.2 量子物理的世界 …… 244
10.2 量子因子分解 …… 246
10.3 量子密钥管理 …… 248
10.3.1 窃听 …… 250
10.3.2 实验验证 …… 252
10.4 本章小结 …… 252
10.5 重要术语 …… 253
习题 …… 253

第1章 密码学概论

1.0 概 述

我们生活在一个令人兴奋、快节奏的世界,任何东西的改变都比不上我们处理信息的方法快。利用 Internet,我们可以利用一些以前想都想不到的方式来访问和使用信息。不用去银行或在取款机前排队等候,我们在家就可以支付账单、签发支票和转账,而且是每周 7 天每天 24 小时都可以。不用出家门就可以申请和接收贷款。可以在 Internet 上买书、食品、礼物以及其他任何东西。不用周末去跳蚤市场,通过网络就可以在任何时间卖东西。我们还可以买卖股票,给他人邮寄信息。随着无线技术的出现,我们用手机在全球任意地方都可以完成所有以上事情,甚至不止这些。

诚然,这些都是令人兴奋的事,但也有不好的一面。同样的技术可以使我们的生活更方便,但若被罪犯利用,就将危及我们的生活。例如,在当今的美国,窃取身份证是增长最快的犯罪之一。它之所以盛行,是因为法律惩罚没有跟上犯罪的步伐,而且这种犯罪很容易实施。这是因为大多数的个人信息缺乏保护。要享受新技术给予的好处,避免陷阱,就必须具有一些保护我们身份和个人信息的方法。如何实现这些正是本书的主题。本书是关于"秘密书写"的,这已有几个世纪的历史了,但现在已经成为了保护和辅助信息技术发展的一个重要作用力。这个领域就称为密码学(cryptography)。

密码学研究的是编码和加密法。David Kahn 在他的被称为"密码学圣经"的著作中是这样定义密码学的:"密码术(cryptology)就是保护。通信对于现代人来说,就好比甲壳对于海龟、墨汁对于乌贼、伪装对于变色龙一样重要。"它已有好几百年的历史了,但仍是年轻、新颖和令人兴奋的。它是一个不断变化且出现新挑战的领域。因此,本书不是一本只让计算机和数学人员感兴趣的干巴巴的教科书。本书同时还介绍了历史、政治学、语言、军事策略,甚至是博弈。本书涵盖了秘密隐藏在我们所生活的世界中的很多知识。

1.1 密 码 学

从某种角度来说,本书是一部肥皂剧。该剧是关于 Alice、Bob 和 Eve 三人的故事。这三个人的名字一直以来就被密码员(cryptographer)用来说明密码学和密码分析法的原理。人们总是这样认为,Alice 和 Bob 相互发送消息,而 Eve 则企图获取 Alice 和 Bob 之间的交谈内容。由于 Alice 和 Bob 都清楚 Eve 的意图,所以他们都尽量防止 Eve 发现他们的消息的内容。这个小型的肥皂剧如图 1.1 所示。

Alice 和 Bob 相互发送的消息称为明文(plaintext),因为这些文字对任何人都是可读的。当他们第一次开始通信时,所发送的明文没有任何保护。但他们很快就发现,如果不对

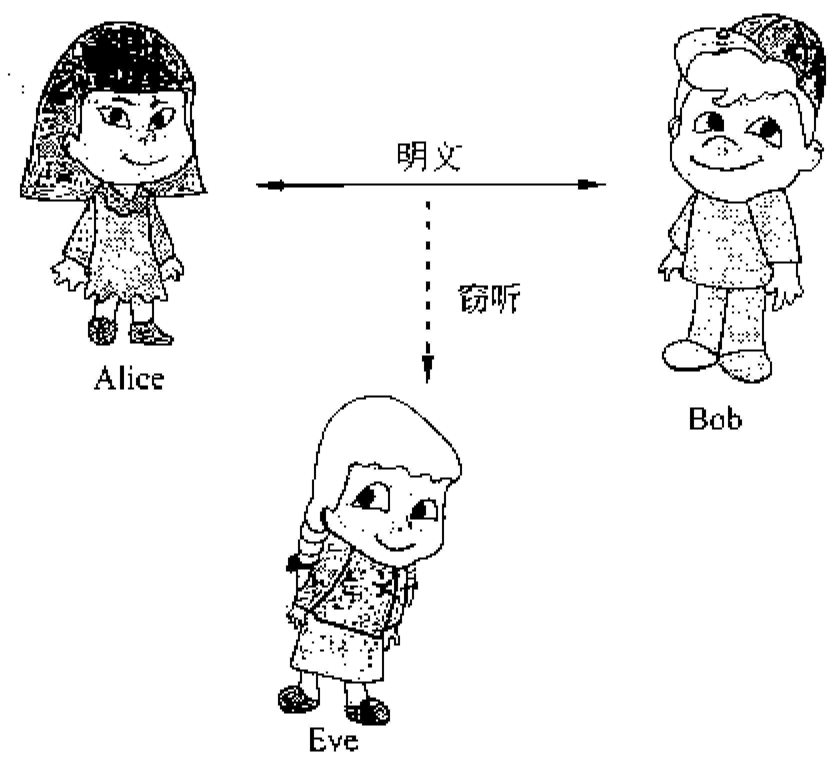


图 1.1 典型的通信模型

这些消息进行任何保护,包括 Eve 在内的任何人都可以阅读它们,因此,为了防止 Eve 阅读他们的明文消息,Alice 和 Bob 决定以某种方式对消息进行保密(hide),这种方式就是他们自己可以还原(recover)明文,而 Eve 则不能。以这种方式隐藏消息从而使其内容保密的过程称为加密(encryption)。经加密后的消息称为密文(ciphertext)。从密文还原明文的过程称为解密(decryption)。加密和解密过程由某种算法决定,并由一个密钥(key)控制,该密钥只有 Alice 和 Bob 才拥有。Alice 用该密钥对其明文加密,然后将密文发送给 Bob。Bob 利用同一密钥将密文解密成明文。即使 Eve 截获了该密文,该密文对她来说也是毫无意义的,因为她没有密钥。这一过程如图 1.2 所示。

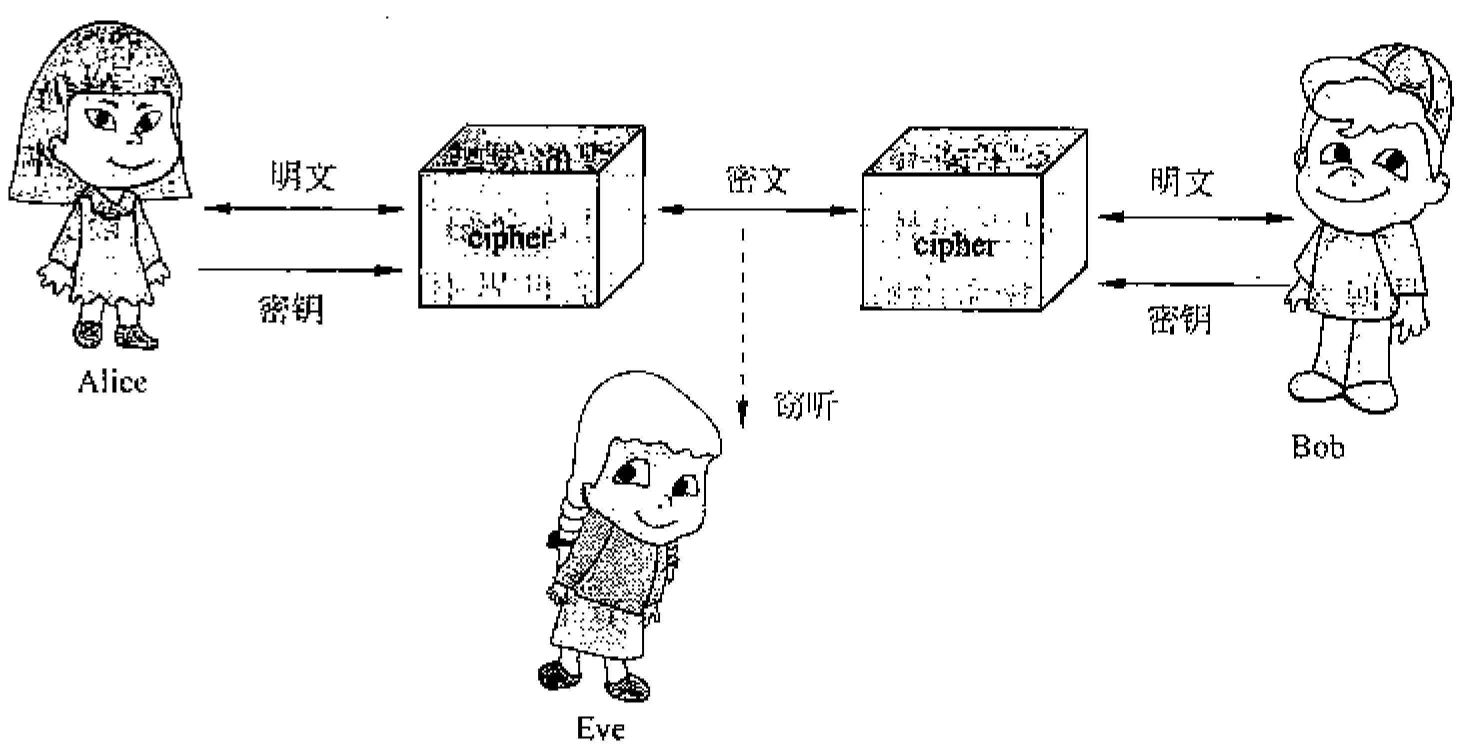


图 1.2 典型的通信模型

Alice 和 Bob 所面临的问题是,Eve 不仅聪明,而且很有毅力。一旦他们使用某种加密方法后,Eve 要破解(break)这种加密法只是时间问题。这就是说,Eve 总是可以不用密钥来还原明文,或从密文中还原密钥。这就迫使 Alice 和 Bob 去尝试更加复杂的加密方法。因此,Alice、Bob 和 Eve 的故事是没有结尾的。Alice 和 Bob 总是企图领先 Eve 一步,而

Eve 则总是以更聪明的方法来破解这种加密法(cipher)。通过本书的学习,你可以顺着这个有趣的故事发展下去,看看 Alice 和 Bob 如何开发新的加密计算,而 Eve 又是如何找到新的工具来破解的。但是,在开始这个故事之前,有必要先来定义一些基本术语。

1.2 重要术语

密码术(cryptology)是构建和分析不同加密—解密方法的科学(某种程序上也可以说是艺术)。该科学实际上是包含两部分。**密码学**(cryptography)是构建功能更强大、更有效的新的加密—解密方法的科学。**密码分析学**(cryptanalysis)则是发现已有加密法的弱点,以便不用密钥就能还原成明文的科学。本书介绍了这两部分内容。你将学习如何保护数据,如何发现当前数据加密法的弱点。对这两个方面的学习,使你更好地了解它们。了解加密的不同方法让你更容易发现某些加密算法的弱点。另外,只有理解加密解析法,才可能判断给定加密法的价值。这就是说,任何一位优秀的**密码师**(cryptologist)必须站在 Eve 的角度,以便判断他们的加密算法的安全性。

首先必须区分的是**编码法**(code)和**加密法**(cipher)这两个术语。因为有时会错误地用它们来描述同一过程。这两者都是用来加密信息的方法,但它们是以完全不同的方式进行的。编码法就是用字、短语或数字来替代明文。例如,"bomb"可能在消息中以数字"1508"的形式出现。从**码文**(codetext)还原明文不存在算法或密钥。生成码文或还原明文需要一本**编码簿**(codebook),它列出了所有数字(或替代字符)和与之相对应的明文字、短语或字母。加密法则是使用算法或密钥来加密信息。

如果一段时间内频繁使用某种编码法,由于编码簿的大小有限,终将给系统的安全性带来威胁。而且,编码的任何改动都需要重新印刷和发布一本新的厚厚的编码簿。潜在的敌人就可能获得发布的编码簿,从而破解编码。而对加密计算密钥的改动,则安全得多。发布一个密钥比邮寄一本厚的编码簿更容易,风险也更小。因此,编码法现今使用很少。但这并不意味着管理多密钥的过程就容易。事实上,在加密法中,密钥管理过程是一个很重要的问题。这些内容将在第 9 章中介绍。

除编码法和加密法外,加密信息的另一种方式是**夹带加密法**(steganography)。该方法是将密文进行隐藏的方式来加密信息的。例如,密文可能夹带在一幅图画或其他消息中。使用不可视墨水是夹带加密法的另一种方式。夹带加密法的使用未能推广是因为密钥问题,但是,随着用计算机进行图像处理的现代技术的应用,这种方法又重现光明了。

1.3 加密法的评价

贯穿本书的主题是如何开发一个好的加密法。通过了解 Alice 和 Bob 是如何领先 Eve 一步,你将学习如何开发一个好的加密法。Eve 对 Alice 与 Bob 的成功破解说明了加密法的一个弱点,这将成为加密法质量的一个测试。每章的末尾都有一个小结,从 Alice、Bob 和 Eve 的角度,概述了一个加密法的设计原理。

但是，任何关于加密法的讨论都必须从加密学的通用原理开始，也就是说，必须假定**偷听者具有加密数据所使用的算法知识**。这就意味着，不要以为加密算法是新的或是对外人是未知的，数据永远都是不安全的。只有加密算法的密钥是安全的，数据才是安全的。永远不要指望敌人不明白你的数据的加密原理。应总是假定他们知道加密算法的详细细节。这就叫作 Kerckhoffs 规律，这也是佛兰德的密码员 Auguste Kerckhoffs 在他 19 世纪的著作 *La Crypthographie Militaire* 中所列举的加密系统必具的六条要求之一。这六条要求现在仍认为是所有加密算法的基础：

(1) 加密系统在实际中应是不可破解的，尽管不是理论上不可破解的。

(2) 破解加密系统应不会打扰通信者。

(3) 密钥应无须做记录即可记住，并容易修改。

(4) 密码应能够用电报来传输。

(5) 设备或文档应一个人即可携带或操作。

(6) 系统应很容易操作，无须掌握一长串的规则或进行专门培训。

20 世纪 40 年代，Claude Shannon 提出了一个常用的评估概念。他认为一个好的加密法应具有**模糊性**(confusion)和**扩散性**(diffusion)。模糊性意味着加密法应隐藏所有局部模式，也就是说，语言的任何识别字符都应变模糊。加密法应将可能导致破解密钥的提示性语言特征进行隐藏。扩散性要求加密法将密文的不同部分进行混合，使任何字符都不在其原来的位置。本书前部分将介绍的很多经典加密法，要么不具备其中的某个特性，要么一个特性也不具备。正是由于未能满足这两个 Shannon 条件，所以它们能被破解。

与其他大多数技术一样，加密系统的评估最终也要落到经济因素上来。一个加密法不必光为了安全而"牢不可破"(而且，它未必就是牢不可破的)。如果获得信息的代价比破解加密的代价更小，就可以称该数据是安全的。或者，如果破解加密需要的时间比信息的有用周期更长，该数据也是安全的。因此，任何加密法的最终安全性是基于这样一个原理：付出大于回报。

1.4 密码分析法

在 Alice 和 Bob 面临创建一个安全的加密法的同时，Eve 也不得不紧跟他们的脚步。Eve 利用所有工具和她所收集的辅助信息，来破解所截获的密文。本书将探究 Eve 的破解过程，让你掌握如何发现加密法的弱点，这就是密码分析的目的。但是，有一点很重要，必须明白，那就是密码分析是一把双刃剑。破解知识可用在好的方面，也可用在坏的方面。学习密码分析法的重要原因是：它是评估新的加密法的必需工具，它在保护国家安全上起着至关重要的作用。另一方面，它也可能成为危害他人隐私的工具。Eve 攻击 Alice 与 Bob 之间的通信可以不是为了任何利益，但她毕竟是有害的。你应确保，你学习密码分析法知识是为了保护自己而不是去危害他人。

Eve 攻击 Alice 与 Bob 的加密法的方法有三种。第一种称为**密文**(ciphertext-only)攻击。当 Eve 获得了 Alice 与 Bob 所传输的全部密文后，她将只凭从密文中获得所有信息来

生成明文。第二种称为**已知明文**(known-plaintext)攻击。在这种情况下，Eve 具有密文和全部(或部分)明文。她可能发现 Alice 和 Bob 总是用同样的句子开始或结束他们之间的消息通信。利用这些信息和密文，Eve 就可能发现密钥。知道一组明文—密文集的密钥，有助于她探查 Alice 与 Bob 的其他通信。第三种称为**选取明文**(chosen-plaintext)攻击。在这种情况下，Eve 试图影响 Alice 与 Bob 之间的消息的特性。她可能给 Alice 一些有趣的信息，她知道 Alice 会将这些信息发送给 Bob。她选择这些信息，这样明文和密文就具有一个很重要的特性，从而有利于密钥的发现。Eve 可以利用这些信息，再加上 Alice 的密文，从而尝试和发现 Alice 的密钥。

显然，密文攻击最为困难，而选取明文攻击最为容易。本书将使用这些攻击方法的一种或多种对所介绍的各种加密法进行攻击。当你学会了破解加密后，应遵守道德和伦理规范。虽然前面已经强调过，但由于这一点太重要了，有必要反复重申。使用密码分析技术破解特定明文只有两个好的原因：国家安全的需要或法律实施的需要。尽管这样，也必须得到特定和法律的许可。另一方面，研究加密系统的弱点也是使用密码分析法的一个很好原因：确保用最健壮、最安全的加密法用于保护敏感信息。

1.5　编码与加密法的历史简介

保护信息的过程具有悠久而迷人的历史。关于加密的历史，有几本很好的书，但关于加密法的经典书则要算由 David Kahn 编写的 *The Codebreakers* 一书了。该书不仅介绍了加密法的历史、技术细节及描述，并介绍了实际的应用。作者建议对密码学领域感兴趣的人都应去读读该书。

在古埃及人的坟墓中发现了早期的编码与加密。其实际的使用已经失传了，但据推测，这些"秘密书法"是为了给墓主的生活增加神秘气氛，从而提高他们的声望。希伯来人开发了三种不同的加密法：atbah、atbash 和 albam，它们都是以替换为基本工作原理的。一个字母表的字母与另一个字母表的字母配对。通过用相配对的字母来替换明文的每个字母，从而生成密文。这种配对关系就相当于密钥，每种加密法都有不同的配对方式。atbah 加密法是先用数字按顺序标出希伯来字母表中的每个字母，就像在英语中将"a"标为 1，"b"标为 2，等等。前 9 个希伯来字符进行配对，这样使它们的值之和为 10；其余的再进行配对，其值之和为 28。消息中的每个明文字母都被配对的字母替换。atbash 加密法是把字母表中的最后一个字母与第一个字母配对，倒数第二个字母与第二个字母配对，以此类推。albam 加密法则是把字母表分割为两部分，再使其两两配对。

据说早期的希腊人使用的一种"秘密书写"方法是，先将奴隶的头剃光，然后将消息刺在头上，等头发长好后，再派他上路。这其实就是夹带加密法的一个示例，因为消息并没有编码，只是进行了隐藏而已。

斯巴达克人是最早将加密技术用于军事消息传递的人之一。他们发明了一种称为 skytale 的工具(实际应拼写为 scytail，其发音为 skytale，因而变成了它的名字)。这种工具是一根有固定面的竿。把一块布或羊皮纸缠在该工具上，再在竿上书写消息。当把布解开后，其上的字母顺序就变乱了(这其实就是换位加密法的一个示例)。要还原明文，须将布缠回到类似的 skytale 工具

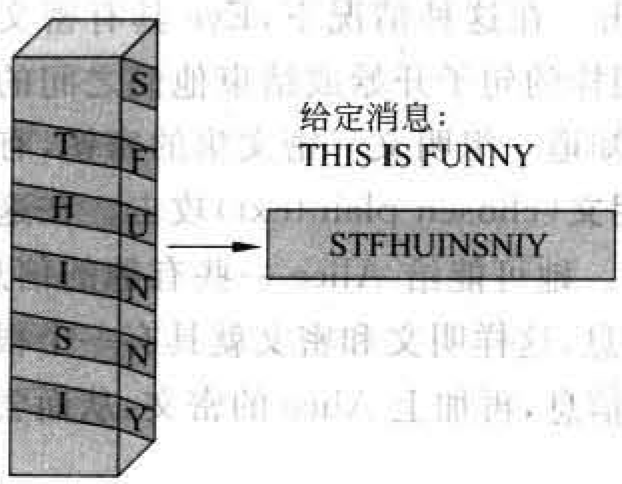


图 1.3 Skytale 加密法

上,如图 1.3 所示。这种对消息的加密方式与希伯来人使用的有着本质的不同。斯巴达克人没有用密文去替换明文,他们只是重新排列了明文字母的顺序。

在 David Kahn 的经典著作 *The Codebreakers* 一书中,他把密码学(即加密与密码分析的组合)归功于阿拉伯人。实际上,也的确是他们创造了"加密法(cipher)"一词。关于密码学的早期主要著作出现在 15 世纪早期由阿拉伯科学家 al-Qalqashandi 完成的百科全书第 14 卷中,它也是介绍密码分析法最早的著作之一。

在欧洲的"黑暗时代",科学艺术,包括密码学,进展甚微。但当欧洲开始从"黑暗时代"中走出来时,编码和加密经历了一个快速发展的阶段。政府成立了大型的间谍网络,并开发了不少加密法以便于他们之间的通信联络,而且这恰好对阅读敌人的加密信息也很有用。因此出现了一个新的政府组织(今天,该组织已成了大多数政府的一个重要部门),称为"保密局(black chamber)"。保密局的任务是截获并解密重要信息。这些组织很成功,并开展了一些很重要的通用解密分析法的初始工作。一个成功的例子是英国的解密部门,在 18 世纪,它的主要工作是阅读美国和欧洲的邮件。

直到 20 世纪美国才设有官方的保密局。这并不意味着美国就不重视加密技术。在 Thomas Jefferson 的很多成就中就有关于功能强大(就当时而言)的加密工具的发明,称为轮加密(wheel cipher)。轮加密使用了一组相互独立的"轮",其上都是以随机顺序刻写的字母表。这些轮可以转动,使明文出现在同一条直线上,而密文则可以从其他任意直线选取,如图 1.4 所示。

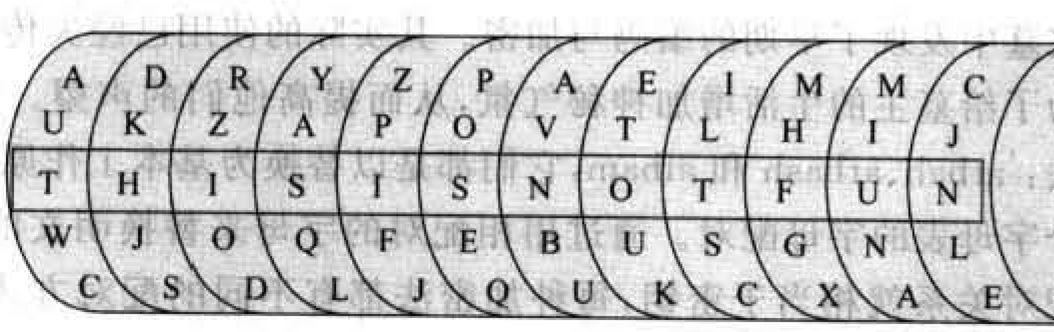


图 1.4 轮加密法示例

第一次世界大战期间,美国终于成立了自己的保密局,称为 MI-8,由 Herbert O. Yardley领导。战争结束后,他继续在纽约市领导该部门。第一次世界大战后,MI-8 非常成功地破获了日本的密码,因此在早期的裁军谈判中,美国处于很有利的位置。所有这些在 1929 年发生了变化,美国的保密局被解散,其任务由陆军和海军承担。

第二次世界大战时期,英国和美国的保密局都成功破获了轴编码法(axis code)。美国政府能阅读日本的大多数密码,而英国(在来自波兰的密码破译家的帮助下)则能阅读德国的密码。不用说,能阅读日本和德国的密码,对联军的胜利起了很大的作用。

今天,美国有了世界上功能最强、最成功的保密局之一,称为美国国家安全局(National Security Agency, NSA),它负责开发新的加密法和对已有加密法的分析。

对于历史上有关加密技术的使用及其影响的真实事件,将在讨论一些经典加密法和现

代加密法的时候介绍。现在，你应该能知道，密码学具有悠久而迷人的历史了吧。

1.6　经典加密法与现代加密法

本书将加密法的内容分为两大类(正如本书的标题所表示的那样)：经典加密法和现代加密法。经典加密法就是以单个字母为作用对象的加密法，而现代加密法则是以明文的二元表示为作用对象。以这种方式描绘其区别，更能清楚地明白经典加密法是有历史(和基础)原因的，而现代加密法更多的是实用。每种类型的加密法又根据生成密文所使用的算法本质，可以进一步细分成如图 1.5 所示。

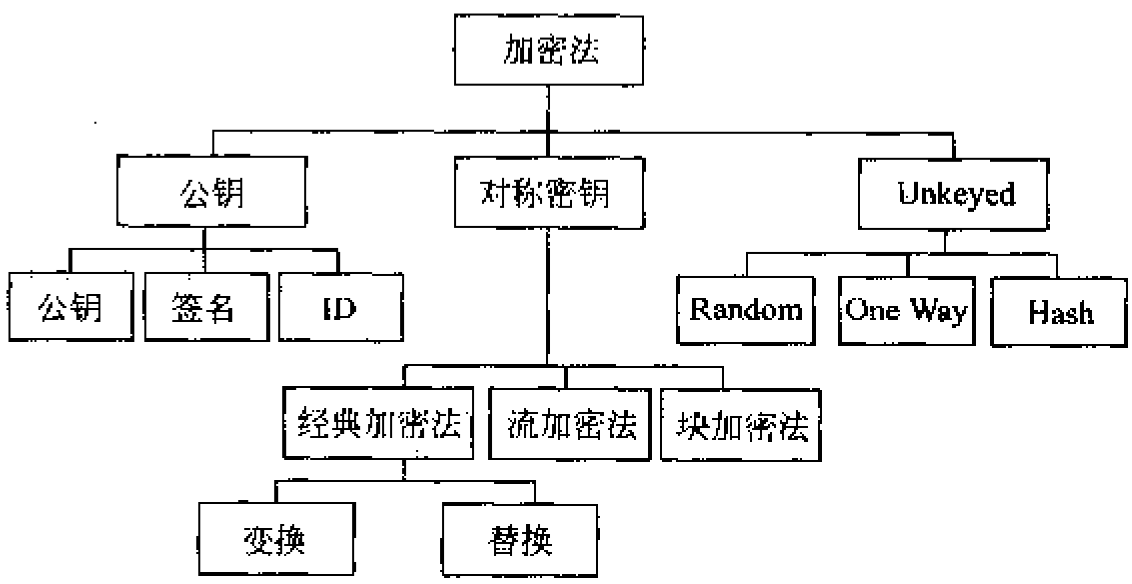


图 1.5　加密法分类

如图 1.5 所示，历史上经典加密法有两种基本类型。一种是替换(substitution)加密法，每个密文字母被其他字母替换，也就是说，每个明文字母"a"可能被密文字母"c"替代。希伯来人所使用的就是这种加密法。另一种是换位(transposition)加密法，明文中的每个字母没有改变，但它们在密文中的位置进行了重新排列(如斯巴达克人使用的 skytale 加密法)，也就是说，单词"next"在密文中可能成了"xent"。

图 1.5 没有显示这两种经典加密法的进一步分类，否则将使该图太复杂了。例如，在经典的替换加密法中，又有两种方法。一种称为单码加密法；另一种称为多码加密法。单码加密法的特征是，每个明文的字母正好映射到一个密文字母，这也就是说，一旦明文的字母"a"被密文的字母"n"替代，那它总是被"n"替代。而在多码加密法中，同一个明文字母可能用多个不同的密文字母来替代。例如，明文字母"a"可能用密文字母"n"、"s"或"y"来替换。显然，多码加密法的生成和破解比单码加密法更困难。但后来发现，即便是多码加密法也有其弱点。

1.7　CAP 软件介绍

本书介绍的很多内容都将用一个称为加密分析程序(CAP)来演示说明。CAP 软件可以从 www.tup.com.cn 下载，它运行在基于 Windows 的 PC 机上。随着不同章节的内容介

绍,你将学习如何使用 CAP 软件,但这只是简短的介绍而已。

CAP 软件是一个 Windows 程序,可以生成和破解密码,它包含了本书所讨论的经典加密法和密码分析技术,以及一些现代的加密系统。它还有一个富有挑战性的博弈游戏,你可以用来测试一下你的加密破解技术。该软件还有一个自动密码分析系统,可以一步一步地指导你如何破解一个加密法。

双击 CAP 软件的图标,可以打开 CAP 主窗口,如图 1.6 所示。从该窗口,可以浏览 CAP 软件的各种不同特性。

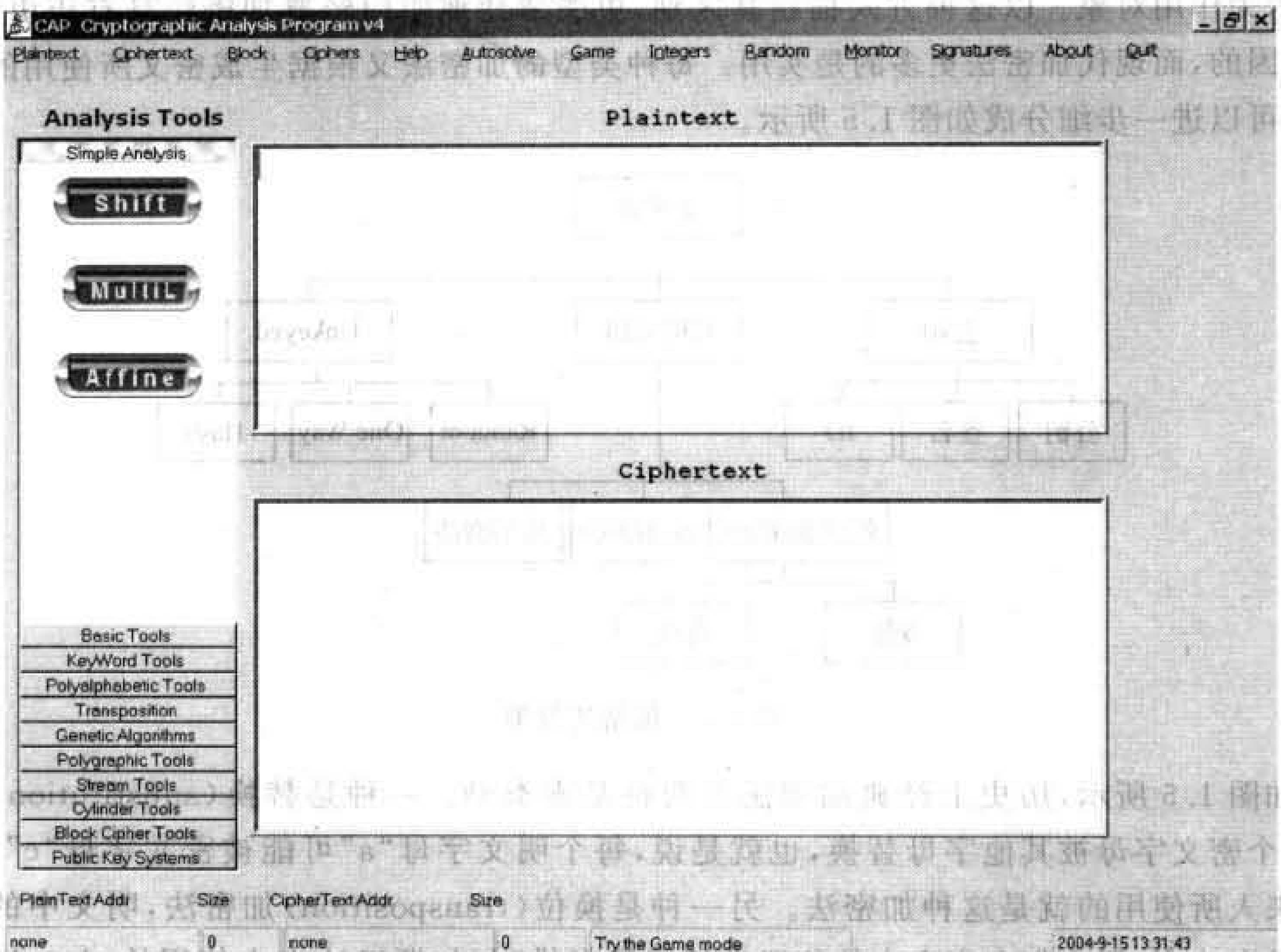


图 1.6 CAP 软件的主窗口

在 CAP 软件的主窗口中,有两个主要区域,分别为 Plaintext 框和 Ciphertext 框。这两个框就像两个小的字处理器,你可以在任意一个中直接输入文字并将它保存,随后还可以在这两个框中将所保存的文件打开。

为了说明 CAP 软件的使用,来看看一个最简单的替换加密法。该加密法称为恺撒加密法,因为朱利叶斯·恺撒使用过它,用来把与高卢进行的战争的消息送回罗马。恺撒使用的密钥是移动 3 位。后来,奥古斯塔斯·恺撒使用的密钥是移动 1 位(即 A 到 B, B 到 C, …, Z 到 A)。首先按顺序写下 26 个字母: ABCDEFGHIJKLMNOPQRSTUVWXYZ。恺撒加密法将把明文中的每个字母用其右边的第 4 个字母替换,也就是说,“a”将被“d”替换,“b”将被“e”替换,以此类推。而后面的字母,如“x”将被“a”替换,“y”将被“b”替换,最后,“z”将被“c”替换。CAP 软件将自动实现这种移位加密。只要在 Plaintext 框中输入明文,从 Cipher 菜单中选取 Simple Shift 菜单项,将出现图 1.7,要求输入移位的个数。恺撒加密法使用的是 3,你也可以选择 1~25 之间的任何数。

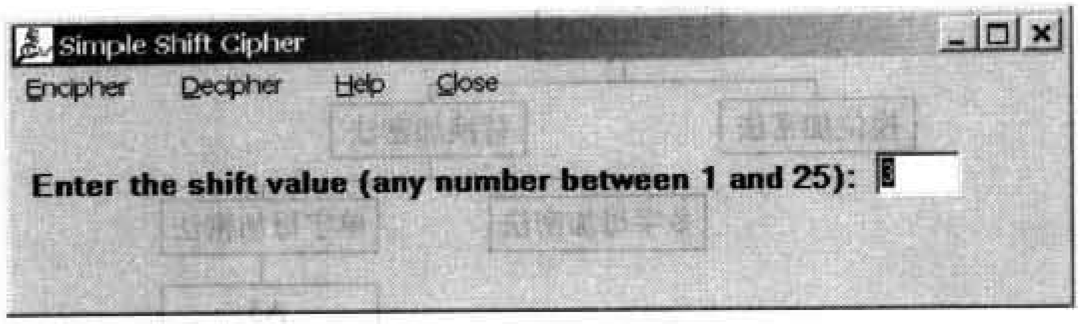


图 1.7 Simple Shift Cipher 窗口

单击 Encipher 按钮即可生成密文。如果输入密文，然后单击 Decipher 按钮即可还原明文。CAP 软件的其他加密法比简单移位加密法要复杂，但在 CAP 中工作方式相同。输入文字(明文或密文)，选取某个加密法，在密钥框中输入密钥，再选取 Encipher 或 Decipher 即可。

CAP 提供的密码分析工具使用也同样简单。例如，要破解简单移位密码(不知道密钥)，只要尝试所有可能的移位数即可。毕竟，该加密法只有 25 种可能的移位数。由于密钥少，破解这种加密法的工作量也很小。简单移位加密法不能满足"付出大于回报"的原则，因此是一种非常脆弱的加密法。你甚至不用动手去分析，因为 CAP 提供了一个工具，它可自动完成其破解。例如，Eve 知道 Alice 和 Bob 首次通信时选用的是移位加密法，但她并不知道选用的是哪个密钥。利用密文攻击法，Eve 先截获从 Bob 发给 Alice 的密文，然后利用 CAP 软件，将密文输入到 Ciphertext 框中，单击 Simple Analysis 中的 Shift 按钮，破解结果将显示在 Shift Cipher Analysis 窗口中，从而可以很快找到明文，知道 Alice 和 Bob 使用的密钥为 3，如图 1.8 所示。

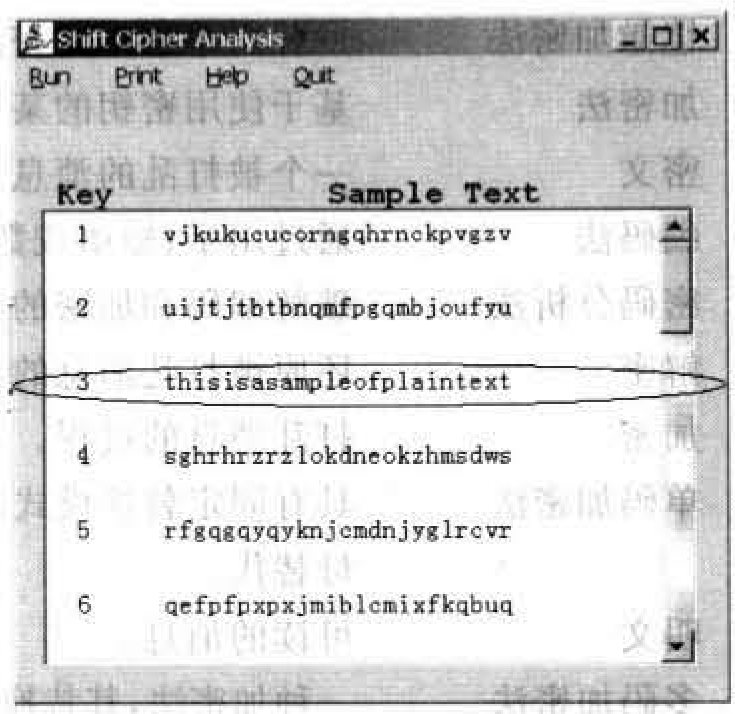


图 1.8 Shift Cipher Analysis 窗口

1.8 本章小结

密码学具有悠久而迷人的历史。从早期的书写方法中，我们可以发现古人以一种能保护隐私和秘密的方式交流思想、想法和打算。我们也可以通过发现他人的秘密来获得优势。这些积极的目的影响着世界的历史发展。

除了历史因素以外，密码术也是令人着迷的，因为它融合了多个科学领域。密码术从语言学、数学、物理学、计算机科学中都吸收了思想和方法。本书将引导你进入这个神秘已久的密码世界。本书将为你解开密码学的历史，帮助你放眼未来。

本章已经介绍了图 1.5 中没有的内容，如果将这些内容加上，则经典加密法的部分将如图 1.9 所示。

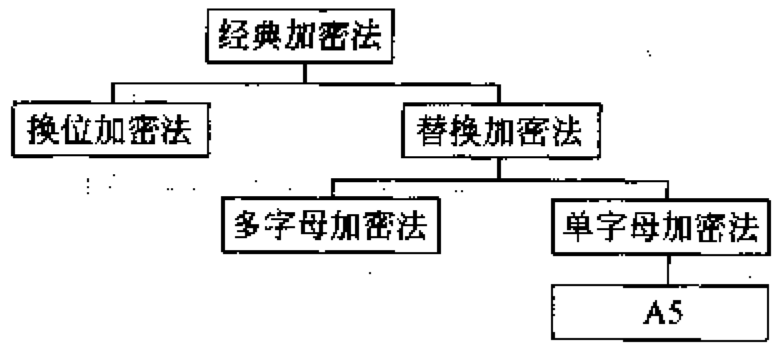


图 1.9 新的加密法结构图

1.9 重要术语

恺撒加密法	一种简单移位加密法，明文中的每个字母被其后的 n 个字母替换。
加密法	基于使用密钥的某种算法，将消息打乱。
密文	一个被打乱的消息。
编码法	通过用字、短语或数字替换明文来打乱信息的一种方法。
密码分析法	破解编码和加密的一门艺术与科学。
解密	还原被打乱消息的过程。
加密	打乱消息的过程。
单码加密法	具有固定替换模式的加密法，即明文的每个字母就由密文的一个字母替代。
明文	可读的消息。
多码加密法	一种加密法，其替换模式是，可以用多个字母来替代同一个明文字母。
夹带法	通过隐藏消息的存在来隐藏消息的方法。
替换加密法	用一个字符替换另一个字符的方法。
换位加密法	重新排列明文中字母位置的加密法。

习 题

1. 何为密码学的首要通用原则？

2. 给定下列明文，请用 CAP 软件生成密钥为 7 的移位加密密文。

The solution to a shift cipher is easy.

3. 利用 CAP 软件找出如下移位密文的密钥和明文。

aopzpzaoljvyyljazvsbapvu.

4. 搜索 Web 网站，找到至少两个专注于密码技术的网站，并就它们的内容写一个总结。

5. 请给出保密局(black chamber)的定义。

6. 编码法与加密法的区别是什么？现在为什么很少用编码法？

7. 根据 Shannon 关于一个好的加密法的两个要求，对移位加密法做一个简短评价。

8. 如果把标准的英语字母表用来生成一个 atbash 类型的密码，替换模式是什么？

9. 由于 Eve 能破解简单移位加密法，Bob 和 Alice 决定使用该加密法两次，也就是说，先以 3 为密钥将每个字母移位以生成密文，再以 5 为密钥对该密文进行移位加密。这样做是一个好主意吗？为什么？

第一部分　经典密码学

第一部分介绍了密码学的基本词汇。对明文、密文、编码法、加密法、替换、移位、多图(polygraphic)及其他重要术语进行了定义。给出了经典加密法的分类，每章都以先简单后复杂的顺序介绍各种加密法，并对其构成和破解分析进行了讨论。

第一部分还介绍了比分类和词汇更重要的内容，即确立了密码学的原理。随着对加密法历史发展及其弱点的介绍，你将明白，耐心和毅力是精通该领域研究的要求。

第 2 章　经典单码加密法

2.0 概　　述

广义地说，经典加密法可定义为不要求用计算机来实现的所有加密算法。这并不是说它不能在计算机上实现(毕竟，本书中的很多经典加密法都已经在 CAP 软件中实现了)，而是因为它可以手工加密和解密文字。大多数经典加密法在计算机普及之前就已经开发出来了，一些加密法现在还被密码爱好者所使用。尽管对任何重要的应用程序，都不推荐使用这些经典加密法，但对这些算法的本质及其特点进行研究，可以为更好地理解现代加密法作铺垫。而且，这些经典加密法比较有趣，它们以一种很简单的方式阐述了那些促进当前密码学发展的概念。

正如第 1 章中所定义的那样，单码加密法是一种替换加密法，其中的每个明文只能被惟一的一个密文字母所替换。例如，在给定的加密法中，明文的字母“a”在密文中可能总是显示为“n”。第 1 章介绍的恺撒加密法就是单码加密法的一个很好范例。如果使用的是移位数 3，则恺撒加密法总是用密文字母“d”替代明文字母“a”，密文字母“e”替代明文字母“b”，密文字母“f”替代明文字母“c”；等等。

本书将介绍历史上曾经使用过的一些经典单码加密法。你将学习如何构造这些加密法，任何使用 CAP 软件中的有关工具破解它们。

2.1 关键词加密法

你可能还记得，在第 1 章中，Alice 和 Bob 开始试图用简单移位加密法来保护他们的通信不被 Eve 破解，但 Eve 使用穷举攻击法(在 CAP 软件的帮助下)可以发现其密钥和明文。没过多久，Alice 和 Bob 就知道了 Eve 经常阅读他们的信息，因此他们决定找一种比简单移位功能更强大的加密法。

Alice 和 Bob 知道，由于简单移位加密法通过尝试所有 25 个密钥就可以很容易破解，所以替代加密法应具有更多的密钥。满足这样要求的一种加密法为关键词加密法。关键词加密法通过以下两步来构造：

(1) 选择一个关键词，如果该关键词有重复的字母，去除除第一次出现之外的所有相同字母。例如，如果选定的关键词为“success”，则使用“suce”。

(2) 将该关键词写在字母表的下方，并用字母表的其他字母按标准的顺序填写余下的空间。

例如，对关键词“magicnet”，这两个字母表将写成为如下形式：

a b c d e f g h i j k l m n o p q r s t u v w x y z
m a g i c n e t b d f h j k l o p q r s u v w x y z

这两个字母表就定义了关键词替换模式。对于上面的关键词，明文字母“a”总是由密文字母“m”替代，“b”则由“a”替代等。在这种情况下，一个典型的明文消息—密文消息对如下所示：

明文 h e l p i a m l o s t
密文 t c h o b m j h l r s

一种改进形式（它看上去是使该加密法更难破解，其实并不然）是，允许关键词从字母表的任意位置开始（即不只是从“a”开始），例如，如果关键词“pacific”从“k”开始，则替换密钥如下所示：

a b c d e f g h i j k l m n o p q r s t u v w x y z
q r s t u v w x y z p a c i f b d e g h j k l m n o

注意，“pacific”的最后两个字母被删掉了，因为“i”和“c”都已经使用过了。在这种情况下，明文—密文对变成了：

明文 h e l p i a m l o s t
密文 x u a b y q c a f g h

这种类型的加密法变种在14世纪的欧洲使用很普遍。到了15世纪，当密码分析技术开始出现时，欧洲政府仍然依赖于关键词加密法，这对保密局来说最高兴不过了。即使是今天，单码—关键词类型的加密法仍在使用，只不过是在密码爱好者中而已（如Bob和Alice）。

CAP软件提供了一个简单的工具来生成各种类型的关键词加密法。运行CAP软件，输入一些明文，然后从Cipher菜单选取Keyword菜单项，将弹出一个Keyword Cipher窗口，如图2.1所示。在输入关键词和开始的位置，然后单击“Set Key”。CAP将显示新的密钥模式，可以进行加密和解密了。如图2.1所示，要对输入的明文进行加密，单击“Encipher”，新生成的密文将自动显示在Ciphertext框中。

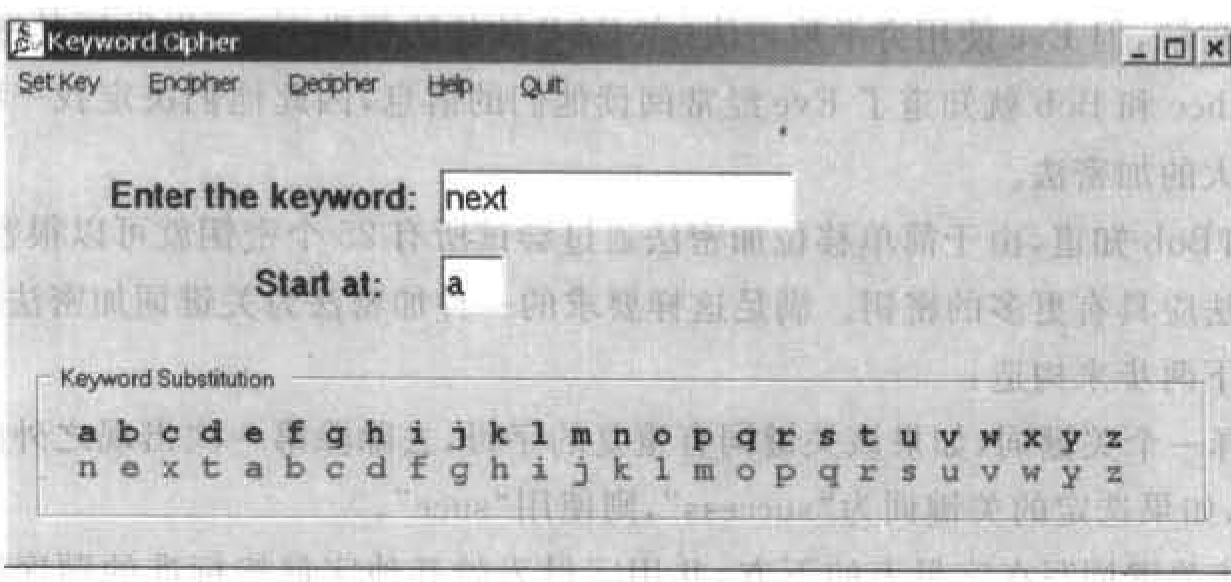


图2.1 CAP软件的Keyword Cipher窗口

2.1.1 关键词加密法的分析法

好了，Alice 和 Bob 开始使用关键词加密法，这将给 Eve 造成一些困难。当 Eve 将截获的密文输入 CAP 软件中，尝试简单移位加密分析时，发现没有产生任何结果。因此，通过其他途径（记住，Alice 和 Bob 不可能有其他不为人知的加密法），Eve 发现 Alice 和 Bob 使用的是关键词加密法。

关键词加密法当然比简单移位加密法功能更强，因为要尝试所有关键词及其位置，显然比尝试所有 25 种移位加密密钥要困难得多。这可能要花费些时间，但是，利用现代的字典攻击法（像用来攻击密码系统一样的攻击法），破解任何关键词加密法都不是很困难。这种攻击法的原理就是计算机用字典中的每个词进行尝试，直至找到一个能破解该加密法的为止。关键点是，今天，用穷举查找攻击法（即尝试所有可能的关键词）来破解关键词加密法也许是可行的，但不可取。假设你想使用比强力攻击法更聪明的方法（每个加密破解员都是如此，总是希望使用尽可能聪明的方法），那么使用一些重要且有趣的加密分析工具，更容易破解关键词加密法。事实上，本书将详细介绍破解关键词加密法的过程，因为它表明和激发了加密分析最基本的使用。这些攻击不仅对破解关键词加密法有用，而且它们也可以用来破解其他的加密法。可能比使用本身更为重要的是，这些工具体现了一种加密法的思考方法。尽管有了计算机的帮助，加密破解员总是希望进行和保持加密分析的脑力训练。学习如何破解关键词加密法，是培养加密破解必备的脑力技术。

假设 Eve 截获了以下密文，并知道是由 Alice 和 Bob 用关键词加密法生成的：

```
GJXXN  GGOTZ  NUCOT  WMOHY  JTKTA  MTXOB  YNFGO  GINUG  JFNZV
QHYNG  NEAJF  HYOTW  GOTHY  NAFZN  FTUIN  ZANFG  NLNFU  TXNXU
FNEJC  INHYA  ZGAEU  TUCQG  OGOTH  JOHOA  TCJXK  HYNUV  OCOHQ
UHCNU  GHHAF  NUZHY  NCUTW  JUWNA  EHYNA  FOWOT  UCHNP  HOGLN
FQZNG  OFUVC  NZJHT  AHNGG  NTHOU  CGJXY  OGHTN  ABNTO  TWGNT
HNTXN  AEBUF  KNFYO  HHGIU  TJUCE  AFHYN  GACJH  OATAE  IOCOH
UFOXO  BYNFG
```

Eve 只知道该加密法是关键词加密法，其目标是发现关键词和明文。这是密文攻击（见第 1 章的定义）的例子。利用其他信息，可以使用其他攻击法。例如，已知明文攻击法就是一种更简单的方法，后面将介绍。

2.1.2 频率信息

要开始破解关键词加密法（或任何加密法），先停下来想想我们已经了解了些什么有用的信息。对于要攻击的关键词加密法，Eve 了解了三个明显但重要的方面：

(1) 明文是标准英语；

(2) 所用加密法为关键词加密法；

(3) 因此，每个明文字母已被惟一的密文替代。

现在，看起来要继续下去很困难，但实际上第3条是Eve破解该加密法的关键所在。**加密分析法最基本的观察是，语言的每个字母都有其自身的特性。这就是说**，如果每个明文“t”被替换为“m”，那么在密文中，“m”将具有“t”的特性。对于任何一个训练有素的观察者来说，字母的特性揭示了其真实面目。字母的这些特性包括：出现的频率、与其他字母的联系、在单词中的位置等。因此，攻击密文的第一步就是尝试确定密文字母的特性，并将它们与相应明文字母的已知特性相匹配。

最容易观察的字母特性是其出现频率。在标准英语中，出现频率最高的是字母“e”，而像“x”和“z”则很少出现。当然，字母的实际出现频率取决于具体文字的类型。例如，科技方面的文字的字母出现频率与文学的有很大的不同。但是，大多数频率分析是基于400个标准英语字母的常见频率表。

字母	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
次数	32	6	12	16	42	8	6	24	26	2	2	14	12	28	32	8	1
字母	R	S	T	U	V	W	X	Y	Z								
次数	26	24	36	12	4	6	2	8	1								

顺序 ETAONIRSHDLUCMPFYWGBVJKQXZ

利用出现频率来破解密码不是最近的发现。早在9世纪，阿拉伯的科学家和哲学家al-Kindi就写下过这种方法。到了15世纪，欧洲的密码学家开始使用频率信息来破解单码加密法。知道了这种技术后，Eve认为，计算所截获的密文中每个字母的出现次数会有用。所计算的结果如下：

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
17	4	13	0	7	17	23	26	5	12	3	2	2	36	25	1	5	0	0	23	20	3	6	9	13	8

开始，Eve试图按标准英语频率表的顺序来匹配字母。加密分析法的一条重要的原则就是做你想做的，她进行了这种尝试，结果如下：

ETAONIRSHDLUCMPFYWGBVJKQXZ
NHOGTUAFCYJXZEWIQBKVLMPDRS

基于这种匹配，每个密文“n”可能是密文“e”，每个密文“h”则可能是明文“t”，等等。要验证这是否正确的惟一方法是在密文中进行这种替换，看是否能组成单词。不幸的是，当Eve这样做后，结果令人失望，因为密文变成了：

OLUUE OOANC EIHAN PJATD…

这比原始的密文好不到哪里去。这对我们来说并不奇怪，因为原始的频率样本和要破解的明文是基于不同字的小集合。因此，实际的频率可能会有些变化（在某个样本中，“i”的频率可能比“a”的要高，尽管通常情况下是要低的），但很少有频率偏差很大的。所有这些说明，频率匹配的工作比最初想象的要困难一些（加密分析法中的很多都是如此）。

因此，可以考虑不去准确的匹配频率，而是考虑将频率分组。标准英语字母分成4个频率组：

高	E T A O N I R S H
中	D L U C M

低	P F Y W G B V
极少	J K Q X Z

字母在分组中的位置可能有些变化，但很少会出现在其他的分组中去。因此，应如何查看单个的频率报告呢？首先，找出高峰和低谷。也就是说，一些字母出现的频率高，而有一些字母的频率则较低，这是单码加密法的表现。第二，查看高频率组与低频率组之间的分隔。不同分组的两个相邻字母的出现频率差应为 2% 左右。第三，最高频率字母通常为"e"，但也可能是"t"或"a"。当然，这只是在具有足够多的密文字母，并能体现"平均"频率分析的情况下才是如此。如果只有少量的字母，那么任何情况都可能出现。

利用这些信息，Eve 回头再去看看原始频率的分布。她首先按频率的高低顺序列出密文字母的出现频率次数：

N	H	O	G	T	U	A	F	C	Y	J	X	Z	E	W	I	Q	B	K	V	L	M	P	D	R	S
36	26	25	23	23	20	17	17	13	13	12	9	8	7	6	5	5	4	3	3	2	2	1	0	0	0

Eve 注意到密文字母"f"和"c"的频率差较大（从 17 到 13），因此，高频率字母很可能是由"n"、"h"、"o"、"g"、"t"、"u"、"a"和"f"表示的。密文字母"n"的频率最大，因而猜测"n"表示的是明文"e"。

频率信息很有用，但要破解这种加密法，还需要其他数据。另一个有用的个性特征是首选关联集，称为双联字母(digram)。每个字母都有这样一个集合，在标准英语文字中，它常与它们一起出现。例如，"th"、"he"和"er"字母对一起出现就很普遍。通过观察密文中的每个字母对，Eve 发现最多的是"hy"、"ot"、"yn"和"nf"。利用这些，Eve 得出这样的结论：密文"hy"很可能是明文"th"，而"yn"可能是"he"。这很好，因为频率信息也表明密文"n"是明文"e"。其他可用来确定密文的数据还有就是，有关明文字母的标准观察。一些比较有用的经验如下：

- 由"r"与不同字母组成的双联比其他任何字母的都多；
- 元音字母"a"、"i"和"o"互不关联(除"io"外)；
- 在元音字母的关联图中，"ea"最多；
- "n"前面的字母 80% 是元音字母；
- "h"经常出现在"e"的前面，几乎从不出现在其后面。

将以上规则应用到密文中，可以得到以下可能的替换：

明文	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
加密	U				N			Y	O					T	A					H						

记住，这只是基于 Eve 当前对密文分析的猜测。一些替换可能是对的，但其他的则可能是错误的。**如果错了，随时准备放弃你的假设，并尝试其他情况。**

一旦确定了一些可能的替换字母后，有几条继续进行的途径。一是查看三联字母(trigram)。另一种是尝试在密文中确定可能的词。例如，图 2.2 就利用了当前的部分结果来查找可能的词。事实上，图 2.2 确定了几个可能的词。例如，在明文的第一行，出现了三联字母"_ith"。这可能是单词"with"，这意味着密文"m"对应的是明文"w"。另一个可能的词出现在第三行的末尾，即字符串"int_ition"。这可能是单词"intuition"，意味着密文"j"对应着明文"u"。

G J X X N G G O T Z N U C O T W M O H Y J T K T A M T X O B Y N F G O G I N U G
E I N E A I N I T H N N O N I H E I E A
J F N Z V Q H Y N G N E A J F H Y O T W G O T H Y N A F Z N F T U I N Z A N F G
E T H E E O T H I N I N T H E O E N A E O E
N L N F U T X N X U F N E J C I N H Y A Z G A E U T U C Q G O G O T H J O H O A
E E A E A E E T H O O A N A I I N T I T I O
T C J X K H Y N U V O C O H Q U H C N U G H H A F N U Z H Y
N T H E A I I T A T E A T T O E

图 2.2 密文中一些可能的单词

现在，Eve 已经非常接近破解该密码了。她将继续查找单词，并且利用字母特性的其他信息，最终发现了关键词和密文。对于这些密码，关键词是“New York City”，起始位置为字母“e”如下所示：

明文	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
密文	u	v	x	z	n	e	w	y	o	r	k	c	i	t	a	b	d	f	g	h	j	l	m	p	q	s

2.1.3 使用 CAP 软件破解多关键词加密法

对喜欢破解的人来说，破解关键词加密法的过程很有趣，但要用手工来完成还是件很枯燥的事。因此，CAP 软件提供了几个工具来自动完成它，包括频率分析系统、特殊字母分析工具、单词查找和关键词工作表。图 2.3 用圆圈标出了 CAP 主窗口中的 Analysis Tools。要使用 CAP 软件的工具，只要从左侧选取一个 Analysis Tools，然后选中某个工具即可。

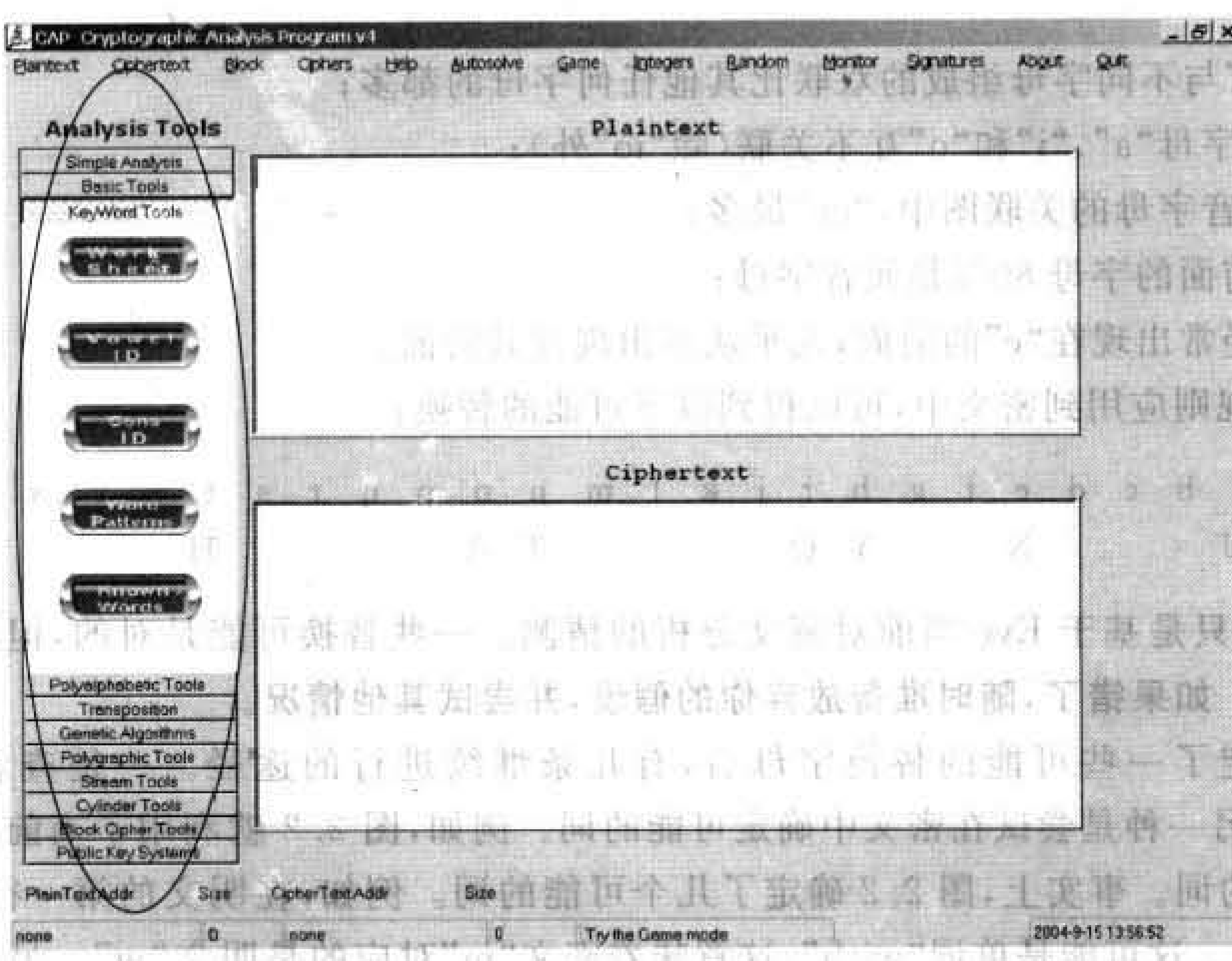


图 2.3 CAP 软件的 Analysis Tools

2.1.3.1 CAP 频率分析

CAP 的 Frequency Analysis 窗口如图 2.4 所示。该窗口中的菜单选项允许用户执行密文或明文字母的单个频率次数、双联字母、三联字母次数。这些报告可以按频率排列，或以图表形式显示。

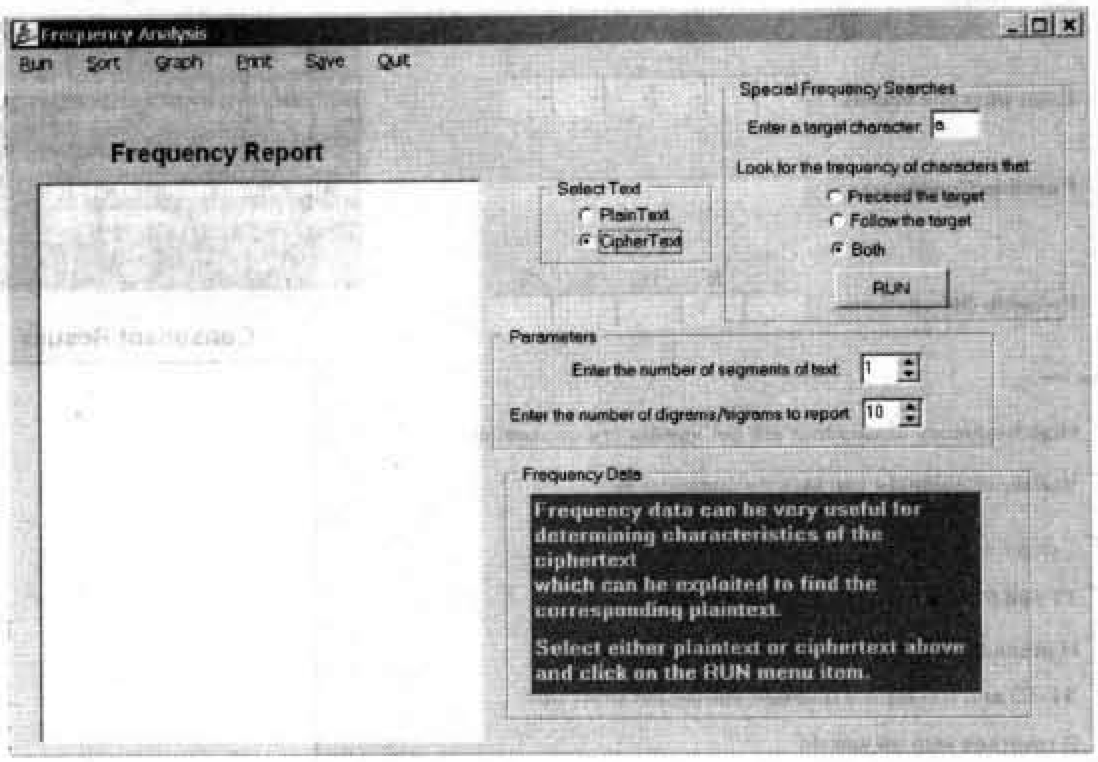


图 2.4 CAP 的 Frequency Analysis 窗口

2.1.3.2 CAP 中的字母确定

CAP 提供了两个用于字母确定的特殊工具，自动应用标准英语的各种规则。vowel ID 工具如图 2.5 所示。要使用该工具，只需选取 Action Steps 中的各个步骤。当 CAP 针对密文查找匹配规则的模式时，会给每个字母赋予一个加权值。具有更大加权值的字母就更可能是元音字母。

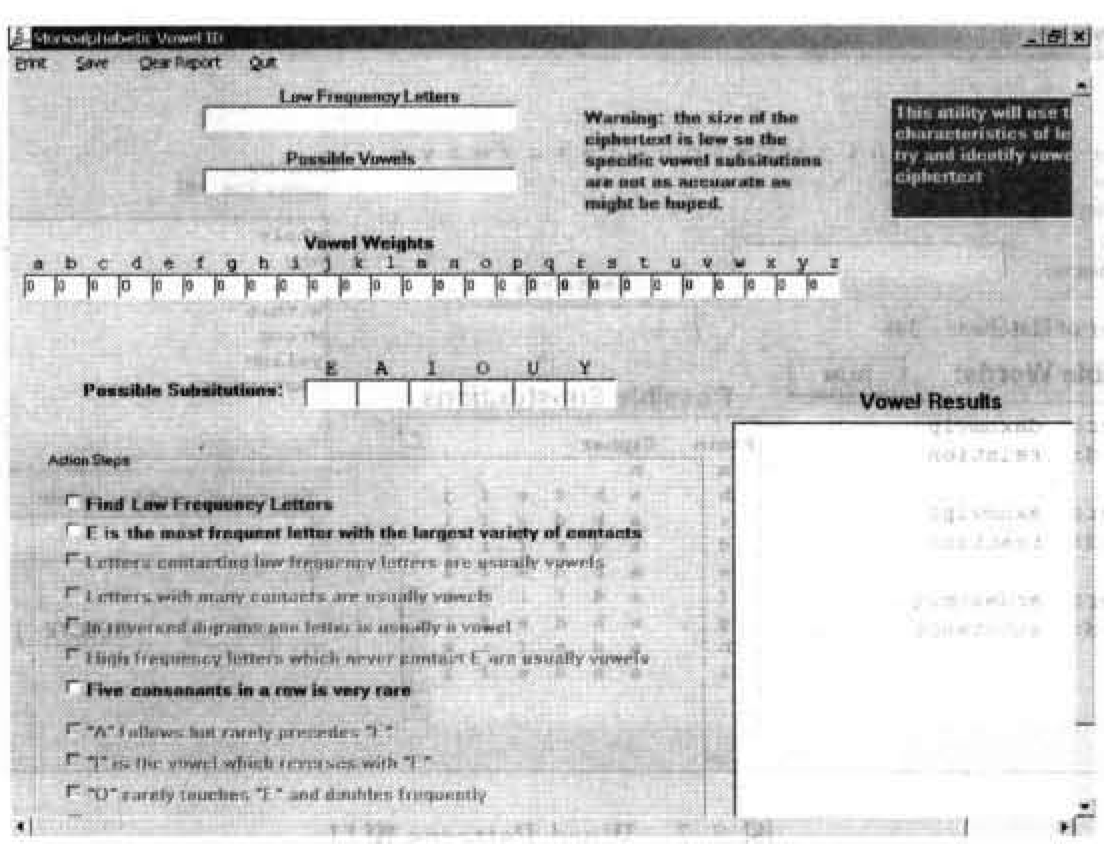


图 2.5 Monoalphabetic Vowel ID 窗口

如图 2.6 所示的 Consonant Identification 窗口的使用与 Vowel ID 窗口类似。它有一系列规则，可用来确定可能的辅音字母。应该注意的是，这两个窗口的运行结果都只是可能的字母替换方式。不同的规则可能产生不同的结果。因此，有必要对每个元音和辅音字母尝试不同的替换方式。

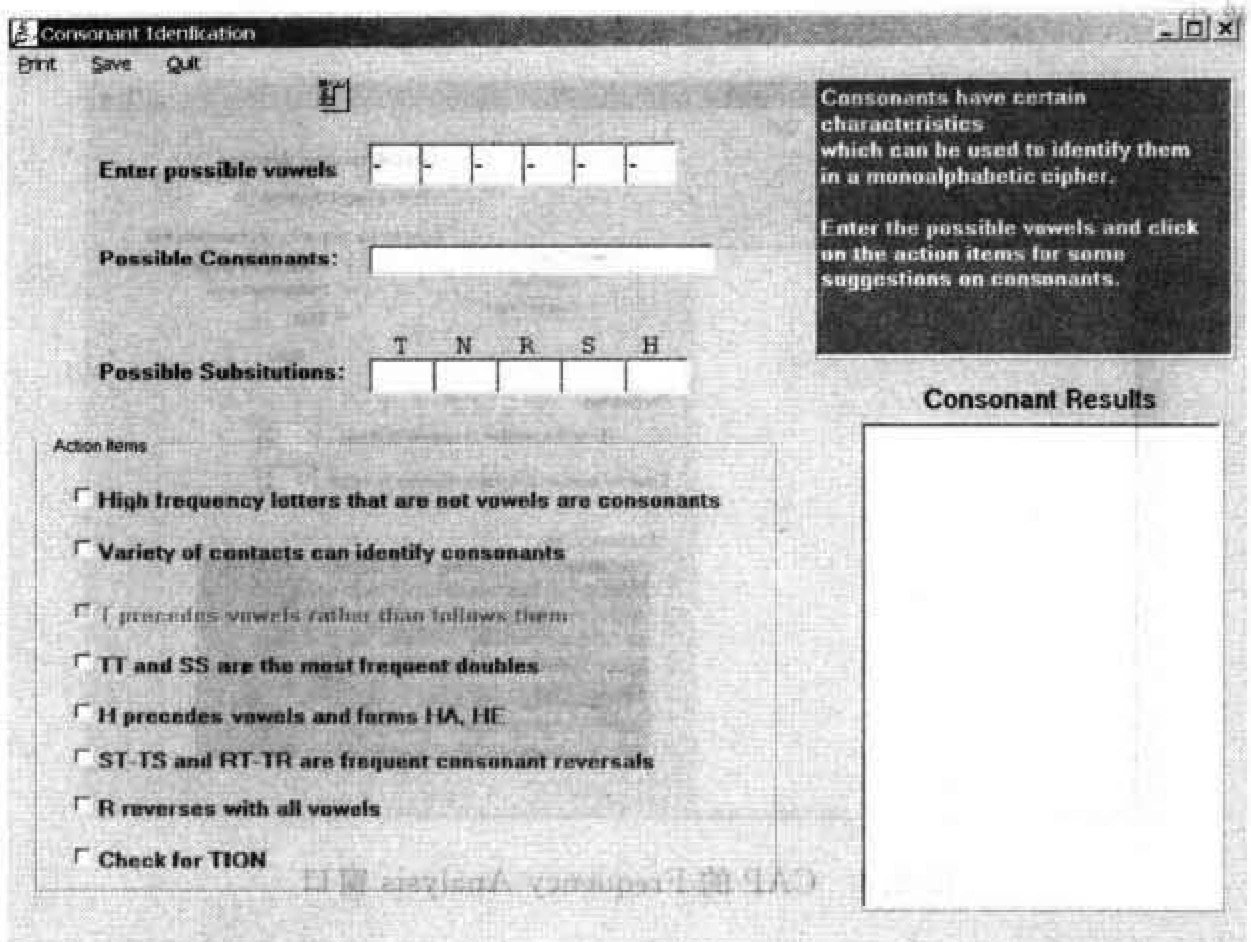


图 2.6 Consonant Identification 窗口

2.1.3.3 CAP 中的单词确定

CAP 还提供了一种帮助查找单词的工具。这是一个自动工具，它利用当前已知的字母替换猜测，查找可能能确定特定单词的模式。Word Patterns 窗口如图 2.7 所示。

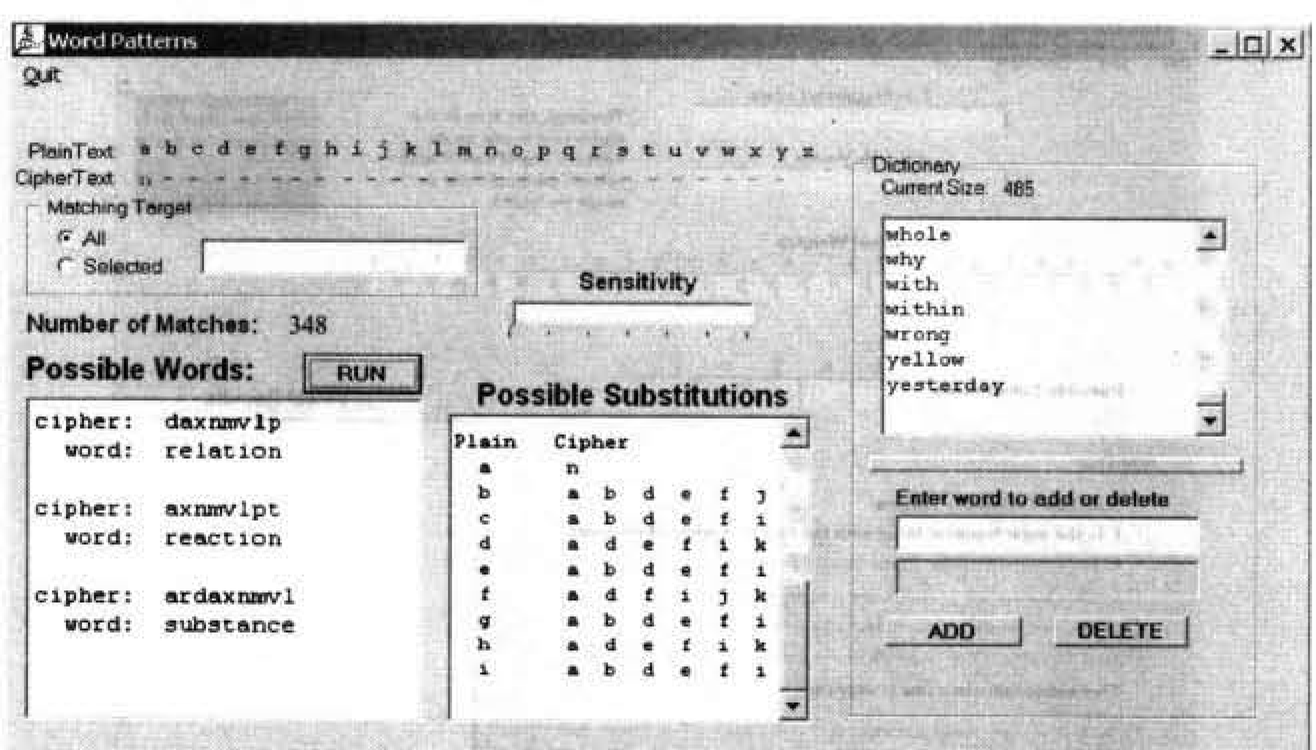


图 2.7 Word Patterns 窗口

这个工具对用密文攻击法来攻击关键词加密法很有用因为它不会假设任何，它只是根据字典中所有单词的模式来查找字母提供模式。CAP 为已知明文攻击法提供了另一种工具。在图 2.8 中，可以查找匹配已知明文单词的可能替换。

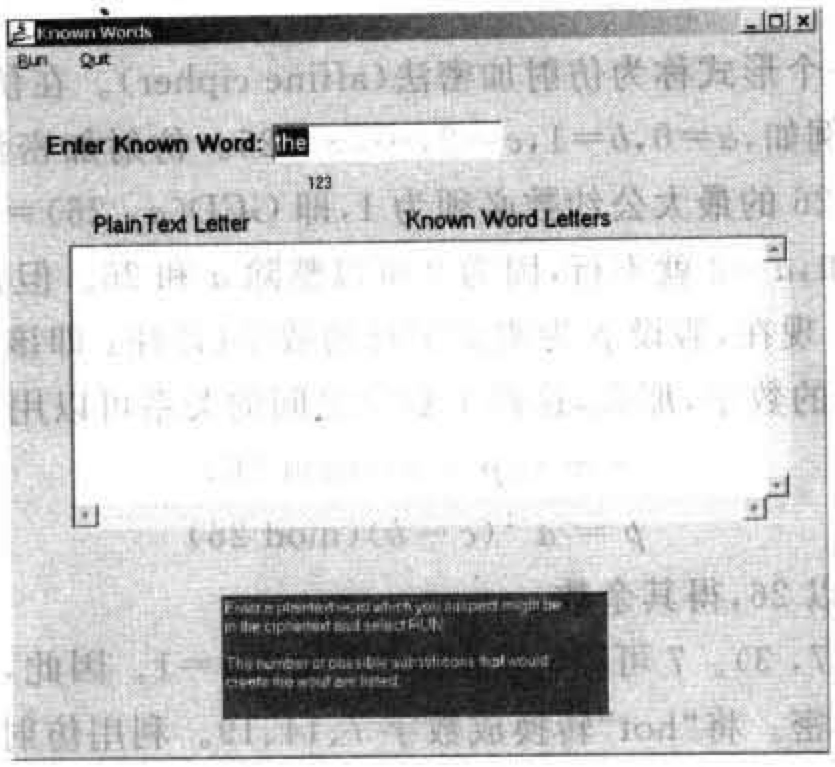


图 2.8　Known Words 窗口

2.1.3.4　关键词工作表

CAP 提供的最后一个工具是工作表，它可以用来尝试不同的字母替换方式，并查看其效果。工作表的一个示例如图 2.9 所示，这可能是破解关键词加密法最有用的工具。它运行你尝试不同的替换方式。例如，在图 2.9 中，出现了单词"test"，表明你的攻击方向是对的。部分单词，如图中的"t--s"可能表示该单词为"this"，因此，明文字母"h"对应密文字母"d"，而"i"则对应"f"。

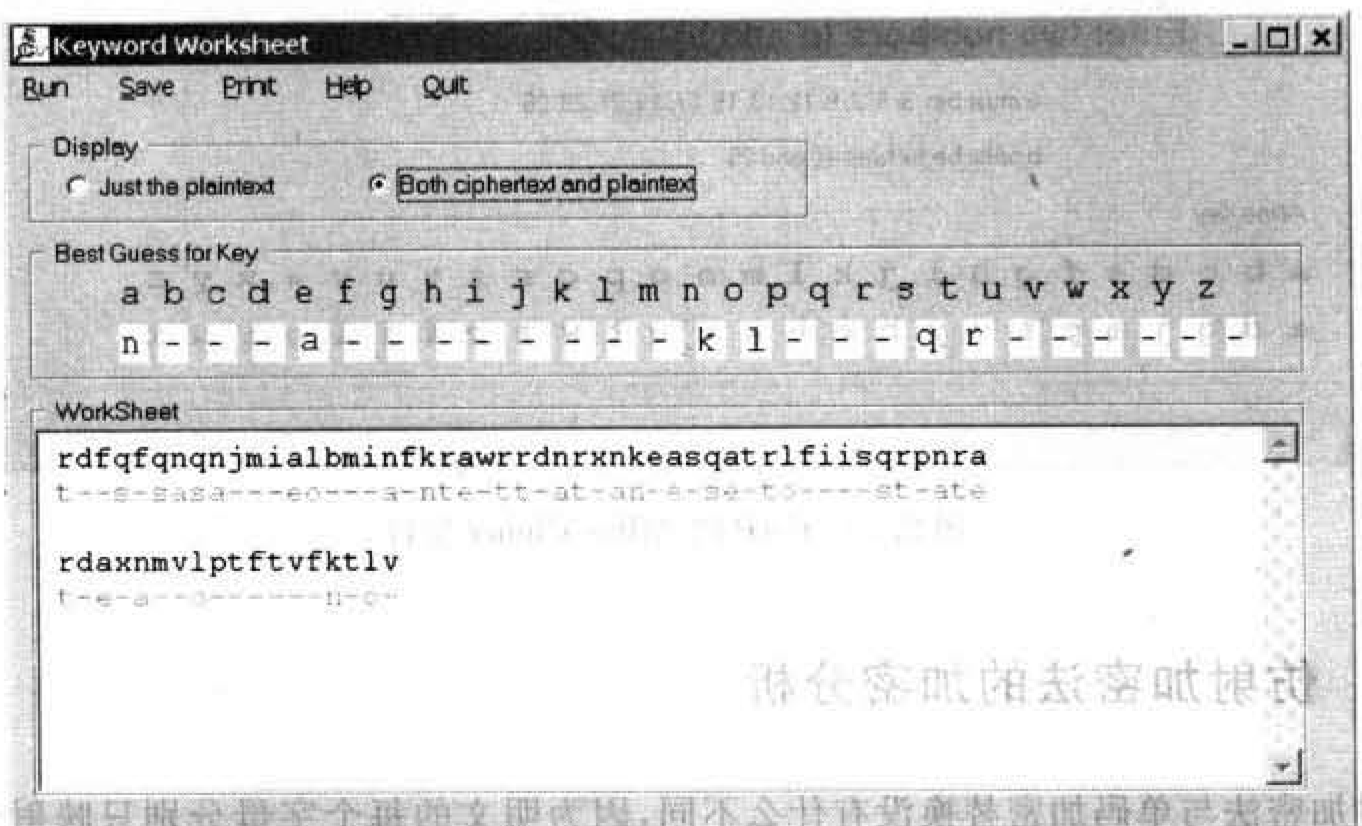


图 2.9　关键词工作表窗口

2.2 仿射加密法

单码加密法的另一个形式称为仿射加密法(affine cipher)。在仿射加密法中,字母表的字母被赋予一个数字,例如,$a=0,b=1,c=2,\cdots,z=25$。仿射加密法的密钥为 0~25 之间的数字对(a, b)。a 与 26 的最大公约数必须为 1,即 $GCD(a, 26)=1$。这就是说,能整除 a 和 26 的数只有 1。例如,$a=2$ 就不行,因为 2 可以整除 a 和 26。但是,$a=5$ 就可以,因为只有 1 才能整除 5 和 26。现在,假设 p 为明文字母的数字(译注:即该字母在字母表中对应的数字),而 c 为密文字母的数字,那么,这两个数字之间的关系可以用如下等式表示:

$$c = (ap + b)(\text{mod } 26)$$

$$p = a^{-1}(c - b)(\text{mod } 26)$$

(mod 26)的操作是,除以 26,得其余数。

例如,选取密钥为(7, 3)。7 可以,因为 GCD(7, 26)=1。因此,利用这个密钥,字“hot”可以使用仿射加密法加密。将“hot”转换成数字 7、14、19。利用仿射等式生成:

$c(H)=(7\times7+3)\text{mod } 26=52 \text{ mod } 26=0$,即为字母“a”。

$c(O)=(7\times14+3)\text{mod } 26=101 \text{ mod } 26=23$,即为字母“x”。

$c(T)=(7\times19+3)\text{mod } 26=136 \text{ mod } 26=6$,即为字母“g”。

这样,对于这个密钥,“hot”变成了“axg”。

CAP 软件可以用仿射加密法直接将明文转换成密文。与 CAP 中的其他加密法一样,选取 Cipher 菜单下的 Affine Cipher 菜单项,弹出如图 2.10 所示的 Affine Cipher 窗口。

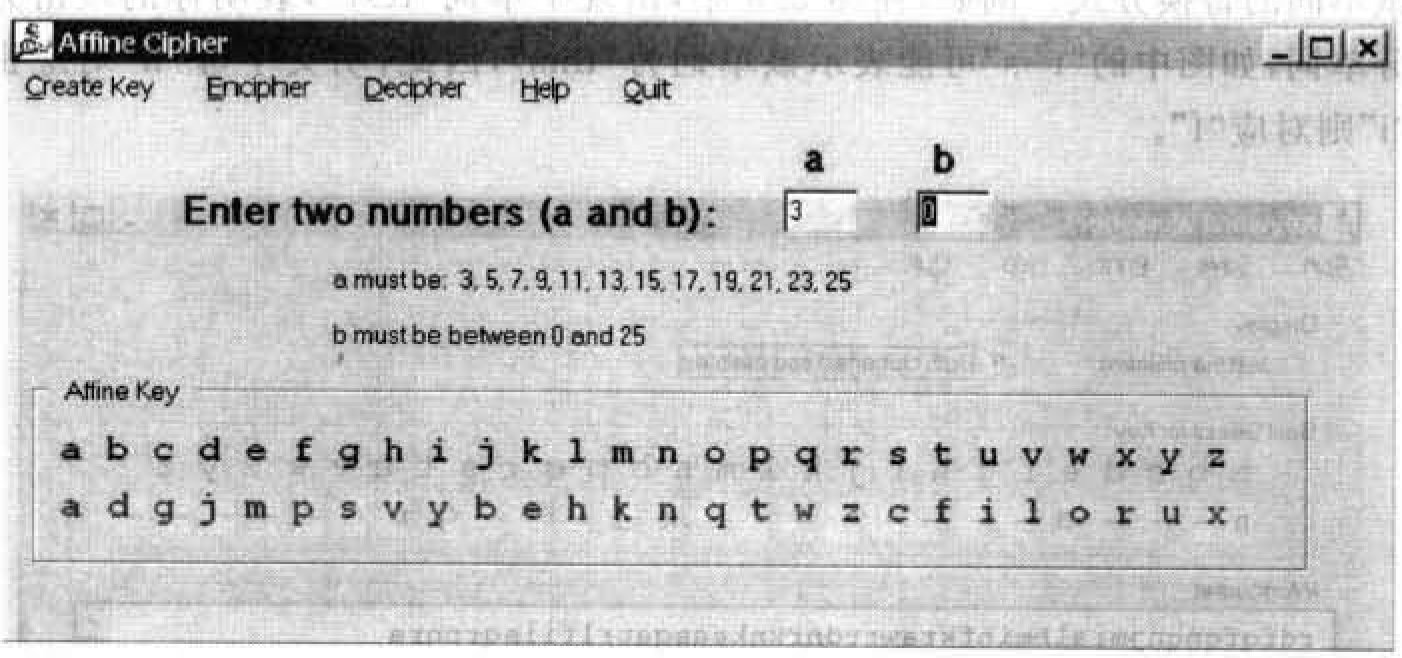


图 2.10 CAP 的 Affine Cipher 窗口

2.2.1 仿射加密法的加密分析

仿射加密法与单码加密替换没有什么不同,因为明文的每个字母分别只映射到一个密文字母。因此,破解关键词加密法的工具同样可以用于仿射加密法。惟一不同的就是密钥替换模式不是基于关键词了。但是,通过给定一个已知的明文攻击,求解出仿射方程式,破

解仿射加密法其实是更容易的。

Alice 和 Bob 发现 Eve 破解了他们的关键词加密法后，决定试图将仿射加密法与单码替换法混淆起来。下一次 Eve 截获了 Alice 与 Bob 的消息后，她可能不会注意到这种变化，因为用于破解关键词加密法的工具对仿射加密法同样有效。一旦她破解了该加密法，也无济于事，因为该关键词不再有任何意义。但是，Alice 和 Bob 永远不要就这样认为 Eve 不会知道他们已经改用仿射加密法了。

一旦 Eve 发现 Alice 和 Bob 换用了仿射加密法，她仍可破解它，或者找出仿射加密法的一个缺陷，从而使其破解工作更容易。

她先进行频率分析，然后使用其他一些关键词破解工具，以便确定至少两个字母的替换。例如，对某个仿射加密密文，确定了明文“e”是由“c”表示，明文“t”是由“f”表示。将这些字母转换成数字，建立仿射加密法方程式：

$2 = a * 4 + b \bmod 26$　　[因为(e—字符 4)被(c—字符 2)取代]

且

$5 = a * 19 + b \bmod 26$　　[因为(t—字符 19)被(f—字符 5)取代]

求解这两个等式，得到 $a=21$ 和 $b=22$。如果不想手工求解这些等式，CAP 提供了一个工具，它可根据图 2.11 所示的两个替换来确定仿射加密法的密钥。但是，应记住的是，该加密法用穷举查找法也是可以破解的，只要尝试所有可能的(a, b)即可。如果字母表的大小扩大，包含字母的大小写，仿射加密法的可用密钥数量就可增大。但总而言之，穷举查找法总是可以破解的。

当 Alice 和 Bob 从关键词加密法改变为仿射加密法时，他们实际上是选择了一种更脆弱的加密法。这并不偶然。通常，看上去功能更强大的加密法往往到头来证明是更脆弱的。

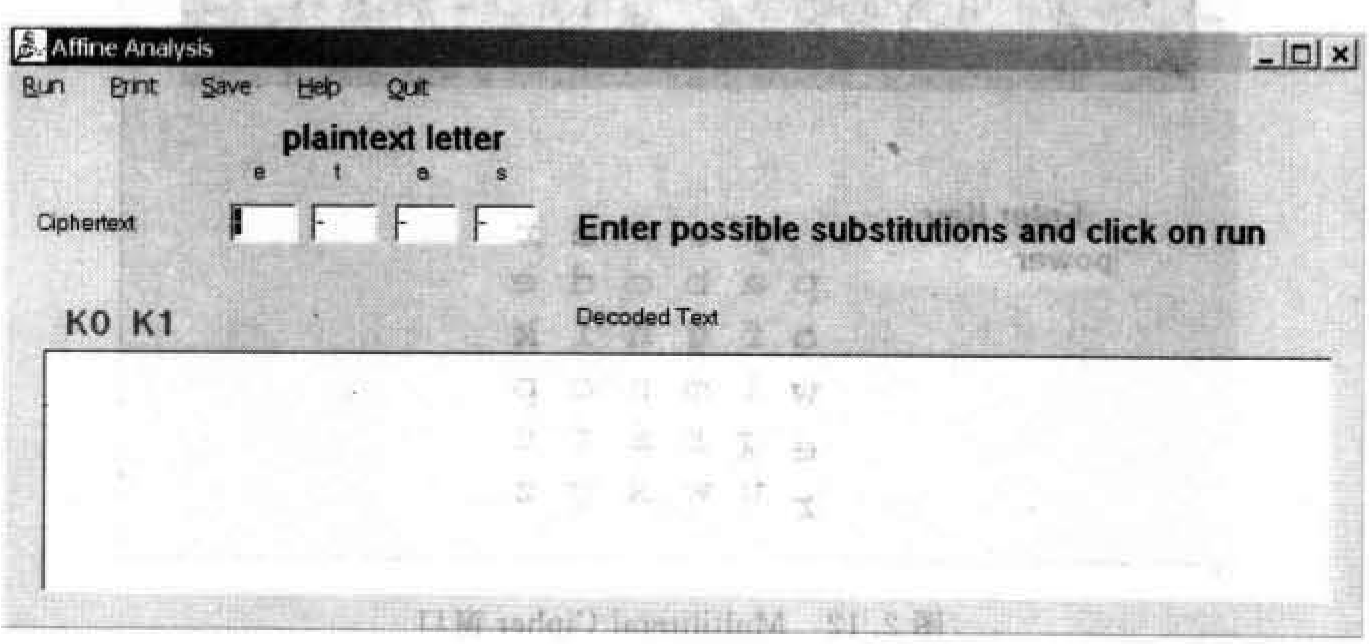


图 2.11　Affine Analysis 窗口

2.3 多文字加密法

多文字加密法是另一种脆弱的单码加密法，之所以值得在这里探讨，是因为它有些特别。它用一对字母来替代每个明文字母。加密后的密文是明文的两倍大，看上去很奇怪的。尽管肯定没有人会使用这种加密法，但它表明了一些有趣的特征，这些特征会在其他一些功

能更强大的加密法中出现。多文字加密法的密钥是一个 5×5 的矩阵。这个矩阵的 5 行和 5 列用含有 5 个字母的关键词来标识。该关键词不能有重复的字母。字母表的每个字母填写在这个矩阵中。当然，矩阵只有 25 个位置，而字母表有 26 个字母，因此字母“i”和“j”占同一个单元。这意味着所有“j”都变成了“i”。但这并没有使加密的明文更难读。例如，如果关键词为 codes，那么密钥矩阵为：

	c	**o**	**d**	**e**	**s**
c	a	b	c	d	e
o	f	g	h	i	k
d	l	m	n	o	p
e	q	r	s	t	u
s	v	w	x	y	z

每个明文由标识该字母的行和列的字母对替代。在上面的情况下，密文对“od”替代了明文“h”。因此，明文“hat”加密后的密文为“od cc ee”。解密过程是反向查找矩阵。CAP 软件提供了一个使用多文字加密法的加密和解密工具。Multiliteral Cipher 窗口如图 2.12 所示。

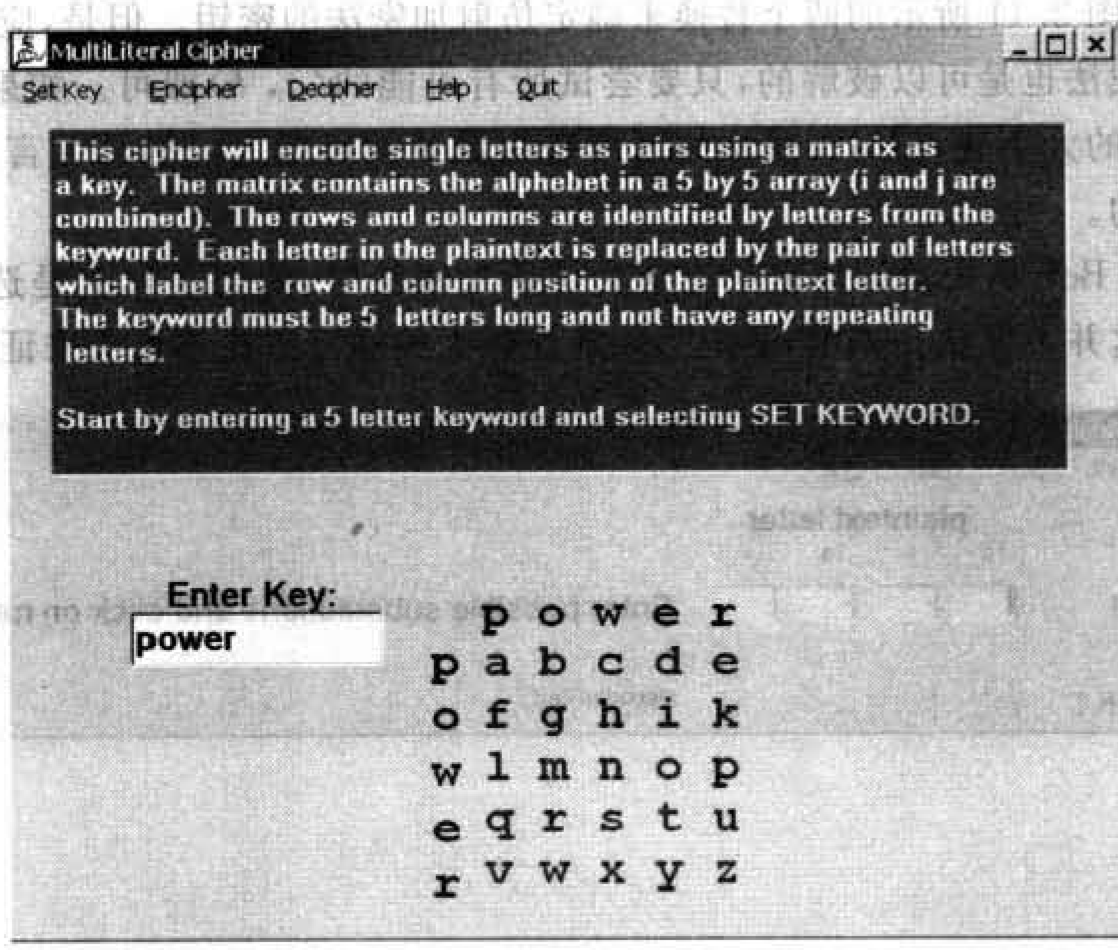


图 2.12 Multiliteral Cipher 窗口

2.3.1 多文字加密法的分析

这个加密法容易破解，因为密码单词只是由 5 个不同的字母组成。因此，密文也只是由 5 个不同的字母组成。Alice 和 Bob 由于使用仿射加密法受到了挫折，决定尝试这种陌生的多文字加密法。不久 Eve 发现了这种变化。现在，Alice 和 Bob 的消息只包含 5 个字母，这是一个致命的疏漏。此时，Eve 所要做的就是将这些字母重新排列成一个合理的单词，而这

很可能就是编码单词。当然,CAP 软件提供了一个多文字分析工具,其窗口如图 2.13 所示。选取 KeyWords 菜单选项就可以尝试这 5 个字母的所有可能的排列。在图 2.13 中,该关键词是"power"。因此,Alice 和 Bob 想安全相互发送消息的企图又失败了。

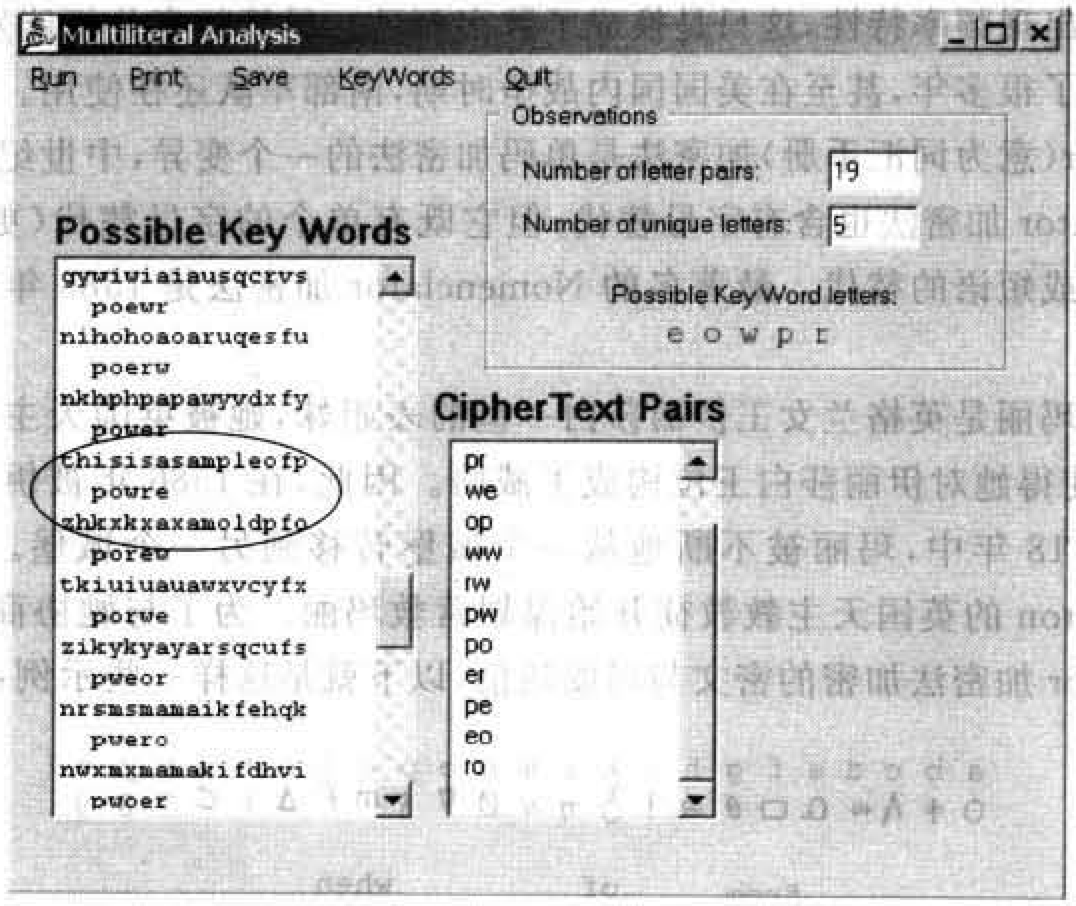


图 2.13 Multiliteral Analysis 窗口

2.4 单码加密法的历史简介

最早的一个单码加密法是希腊作家 Polybius 在大约公元前 200 年发明的。该加密法称为 Polybius 方格,因为它将字母表的字母填充在一个正方形中,并给行和列加编号。每个字母由对应的行号和列号来替代。对英语字母表来说,Polybius 方格如下所示:

	1	2	3	4	5
1	a	b	c	d	e
2	f	g	h	i	k
3	l	m	n	o	p
4	q	r	s	t	u
5	v	w	x	y	z

注意,由于只有 25 个位置,所以去除了字母"j",这与多码加密法类似。因为"j"是低出现频率字母,在明文中出现的时候少,它可用字母"i"来替代而不会影响文字的可读性。与简单移位加密法不同,Polybius 方格的概念沿用了很多个世纪。在下一章中将会介绍,它是 Nihilist 加密法的起源。

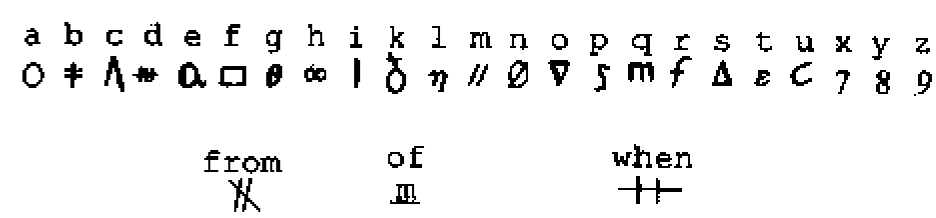
Argentis 家族(以 Giovanni 和他的侄子 Matteo 为首)是经验丰富的密码学家,15 世纪末 16 世纪初期为教皇工作。显然,是他们最早使用关键词加密法的。但是,与 Polybius 方格一样,是用数字来替换字母的。一个密钥如下所示:

p	i	e	t	r	o	a	b	c	d	f	g	h	l	m	n	q	s	u	z
10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

这两种加密法都可用标准关键词攻击法来破解。在密文中使用数字(而不是字母)也不能隐藏其潜在的出现频率特性,这只是换成了数字而已。尽管频率分析能发现其弱点,单码加密法还是沿用了很多年,甚至在美国国内战争时期,南部军队还在使用。

Nomenclator(意为词汇手册)加密法是单码加密法的一个变异,中世纪时期被欧洲广泛使用。Nomenclator 加密法也含有字母替代,但它既有单个的字母替代(更复杂,第 3 章将介绍),也有单词或短语的替代。最著名的 Nomenclator 加密法是 1586 年由苏格兰女王玛丽使用的。

苏格兰女王玛丽是英格兰女王伊丽莎白一世的表姐妹,她被英国天主教认为是英格兰世袭的女王,这使得她对伊丽莎白王位构成了威胁。因此,在 1586 年被伊丽莎白女王打入囚牢。在以后的 18 年中,玛丽被不断地从一个城堡转移到另一个城堡。最后,一个名为 Anthony Babington 的英国天主教教徒开始谋划营救玛丽。为了与她协商密谋,Babington 用经 Nomenclator 加密法加密的密文与玛丽通信,以下就是这样一些示例:



Nomenclator 加密法还可以包含空值或符号,用来代表一些单词,如 not、so、and、but 等。很长一断时间内是由 Gilbert Gifford 将信件偷带给玛丽,他负责将 Babington 的加密文字给玛丽,并取回玛丽的回复信件。对 Babington 和玛丽很不幸的是,Gifford 是双重身份,他实际上是为伊丽莎白女王的首相 Francis Walsingham 工作的。Walsingham 在伦敦领导一个加密研究所。他的主要密码专家是 Thomas Phelippes。Phelippes 没用多久就破解了玛丽的 Nomenclator 加密法。不久,Walsingham 就可以阅读密谋营救玛丽、暗杀伊丽莎白女王、将玛丽推上王位的全部内容。当他有足够的证据让伊丽莎白女王处死她的表姐妹时,Walsingham 想得到更多,他想知道刺客的名字。他所做的已经不是密码学的范畴了,这即使在今天仍是一个重要的问题(第 9 章将详细介绍)。他截获了玛丽的一个消息,并添加一个由 Nomenclator 加密法加密的附文。该附文询问其他密谋者的名字。当 Babington 收到这个经修改过的消息时,他根本没有表示怀疑,毕竟,这是加密过的,因此肯定是来自玛丽的。这个修改过的消息送出去不久,Babington 和玛丽就被逮捕,并指控犯有反叛罪。玛丽的辩护是她对 Babington 的密谋一无所知。她认为她怎么能从监狱中指挥一个刺客呢?但那些加密文字揭示了她的谎言,她于 1587 年 2 月 8 日被处决。

2.5 本章小结

当然,人们还开发和使用过很多其他的单码加密法。但是,它们都具有本章介绍的加密法相同的弱点。在使用这 4 种单码加密法都失败后,Alice 和 Bob 认为该继续努力,以找到一个功能更强大的加密法。他们的故事将在下一章中继续。现在,有必要考虑一下,Alice、

Bob 和 Eve 在保护和破解数据的斗争中，都学到了些什么。

Eve 发现了密文频率分析的强大功能。利用一些有创造性的想法，再加上有关单个、两个、三个字母组合的频率数据，她就可以确定 Alice 和 Bob 在单码加密法中使用的替换方式。她还知道，即使是有功能强大的工具帮忙，要破解某些加密法还需要很大的耐心，而有些则很容易就破解了。耐心和尝试不同想法的欲望将有助于去完成更富挑战性的加密法。

Alice 和 Bob 明白了单码加密法太脆弱了。他们需要更复杂的加密法。由于他们现在已经领教了频率分析的功能，他们将寻找能挫败频率攻击的加密法。

应该注意的是，使用数字或其他符号来替换明文字母的单码加密法并不比纯粹的关键词加密法强多少。因为这些数字或符号还是会具有明文字母的特征，熟练的密码分析员仍可以破解它。

最新的加密法结构图如图 2.14 所示。

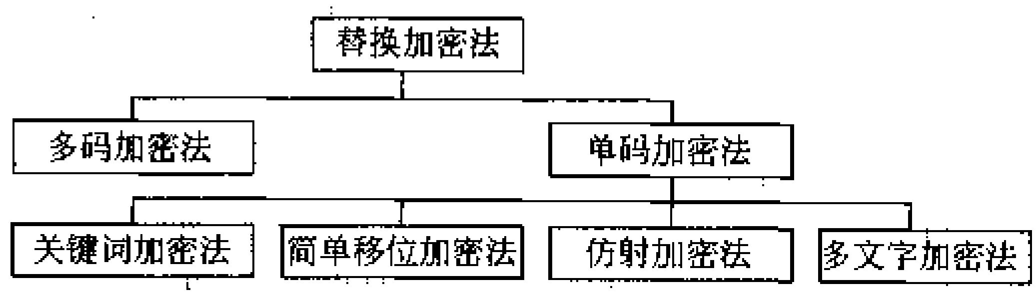


图 2.14　替换加密法结构图

2.6 重要术语

连字	文字中一对相临近的字母。
频率分析法	将明文字母的出现频率与密文字母的频率相比较的过程。
GCD	最大公约数，即，给定两个数，GCD 就是能整除这两个数的最大数字。
单码加密法	明文的每个字母只能被密文的一个字母替代的加密法。

习　题

1. Alice 发送以下消息给 Bob：

I think Eve has discovered how to break this latest cipher. She mentioned something to me yesterday that indicated that she might have read our last message. I recommend that we try something different.

(a) Alice 从字母“n”开始使用关键词为“please”的关键词加密法，重新生成其原始密文；

(b) 就这些密文进行单字母频率分析，从分析结果中确定哪个是字母“e”，并将结果与已知的实际情况进行比较。

2. 如果 Alice 使用密钥为(5, 11)的仿射加密法，请重做问题 1。

3. Eve 截获了 Alice 给 Bob 的如下消息：

qihihpkgqigqukybmpqiyqukfigqubgqukktmex
fimgbgobtftymtbnqutqkskmkftbgpigkimfimkpqkjp
tuktniyrppukpkkfpqihkthekqiorkppvutqvktmk
nibgohkyimkvkkskgdgivbquivatgvkkskmvbgbgqubp
htqqekiyvbqphrqbvbeegiqobskrjuijkvkvbeecrpq
utskqiqmxutmnkmqutgkskmqidkkjirmfkpptokpptyk
tgnpirgn

她知道这些消息是用关键词加密法加密的，但关键词显然不是"please"了。请找出该关键词和明文。

4. Eve 截获了另一些消息，这次是从 Bob 发给 Alice 的。她知道他们还是用的关键词加密法，但是又更换了关键词。利用从上一问题中知道的消息和 CAP 工具，找出下列消息的关键词和明文。

tmhjf	pasal	tmltd	abfrs	tmbtd	akmlt	dfltd	ahbta	rlmml
emlmt	oahbr	hfehb	tarhj	javaf	smljy	dukhl	sdaph	llmtp
mltfl	uatmo	asmjn	piyfh	ksura	wawfj	jbfle	hiayt	dhtsd
aphll	mtora	himrh	ttdav	aryja	hstwa	wfjjp	mkaun	wftdt
danar	baptp	fndar	systa	khlel	avard	hvatm	wmrry	homut
darhc	hflom	o						

5. Alice 和 Bob 发现 Eve 破解了他们的关键词加密法，因而决定同时使用两个关键词，也就是说，用一个关键词将明文加密，然后用另一个不同的关键词对密文进行再次加密。这样真的有用吗？为什么？

6. 请填写以下内容：

加密法	操作总结	弱点总结
关键词加密法		
仿射加密法		
多文字加密法		

7. 用关键词为 guide 的多文字加密法将下列文字加密，然后进行频率分析：

A multiliteral cipher is easy to break and it gives itself away by its very structure.

从分析结果中你能得出什么结论？

8. 确定下列密文的加密类型，并找出其明文。

ccicooiofcoioefocficifoficoffcociciiccccfifciifeccfifcciifceio
ccfciofcccoecfooiooifciefioecffcciifceoififioeioieifooecofoeef
oecifiicceiooccccfecciicifoeccfifciiffoeiocioe

9. 找出下列仿射加密法的明文：

xdftv pxkdh hxkxq qrkhp rcihu rdhxd rhumt dtseh mixkx

dmxky xuyjh fntuy preih utkph ftvix ehdtb hrkrm rxsrk
qtubx mrtkt kmihc tddrg shdvg dmrmv mrtkd

10. Eve 截获了 Alice 与 Bob 之间的如下消息：

rdeqe qftqr cntem gjkru rkoui ejpyk trdcr oubeq rocre kjsko rduju xrqui uqruo
crqmd kkheq mkiej btlqk kjdcv uyktp umcpu pwdem dmhcq quqyk twehh aurcg
ejbyu resqk huriu gjkwq kemcj qurtl iymhc qqqmd upthu cjpwu mcjrc gucrh ucqrk
jumhc qqrkb urduo ptoej brduq loejb huriu gjkww dcreq bkejb kjcqq kkjcq lkqqe
ahu

她认为这些消息是用关键词加密法加密的，并认为"registration"一词出现在了消息的某个地方。利用这些信息找出明文。

11. Eve 还原了关键词加密法的下列部分密钥：

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
s	e			i				b	d	f	g	h		k	l	m	n								z

你能完成该密钥的其余部分吗？

第3章　经典多码加密法

3.0　概　　述

正如第1章中所定义的，多码加密法是一种替换加密法，其中的每个明文字母可以用密文中的多个字母来替代，而每个密文字母也可以表示多个明文字母。例如，明文"e"可能在密文中有时出现为"f"，有时又出现为"m"；密文字母"s"有时可以表示明文"g"，有时又可以表示明文"c"。多码加密法很容易与同音异义(homophonic)加密法混淆。在同音异义加密法中，密文文字是由符号(也可能是数字)集组成，每个明文字母对应多个密文符号。同音异义加密法与多码加密法的区别是，每个同音异义符号仅仅表示一个明文字母。例如，在同音异义加密法中，字母"a"可能被数字1、27和56替代，而字母"b"可能被数字2、28、57替代。在密文中，27总是代表"a"。但在多码加密法中则不同。同音异义加密法并不比传统的单码加密法功能强多少。由于某些符号集总是表示特定的明文字符，所以仍然能体现该字符的某些特征。

15世纪的建筑师Leon Battista Alberti被认为是第一个使用多码技术的人。他设计了一个加密转盘，其外圈部分可绕着内圈转动。内圈上有明文字母表，而外圈上有密文字母表。加密方法是，先设定外圈的初始位置从而确定替换模式。当一些字母被加密后，随着外圈的转动，替换模式将发生变化。这种方法将导致不断替换模式的改变。

产生于15世纪的多码加密法在中世纪时代受到了**词汇手册**(nomenclator)加密法的挑战，该加密法在当时已经成为了加密标准。词汇手册介乎编码法与加密法之间。一些字和短语由编码来替代，而单个的字母则选择一个或多个符号来替代。西班牙国王Philip在1556年就曾采用过一种很复杂的词汇手册加密法。该方法具有一种多码替换模式，对每个辅音字母有两个替换选择，对每个元音则有三个替换选择。编码部分则为常用字和标题提供了两三组替换选择。常用的两个和三个字母的词可以被一个符号或一个数字替代。

多码加密法的开发是为了用来对付频率分析工具的，频率分析工具对破解单码加密法很成功。多码加密法的目的是，通过用多个密文字母来替换同一个明文字母，从而消除字符的特性。结果是，单个、两个和三个频率不再能体现实际的明文字母了。与第2章讲解单码加密法一样，本章将介绍一些比较著名的经典多码加密法，并开发一些新的分析工具。

3.1　Vigenere加密法

Vigenere加密法是一个很著名的多码加密法，它具有一个很有趣的历史。Vigenere加密法实际上是**自动密钥**(autokey)加密法的一个简化形式(自动密钥的概念将在本章的后面介绍)，由Blaise de Vigenere于16世纪中期开发的。该加密法的历史一直失传，直到19世

纪后期才被重新发现。由于某些原因，那时的密码学家将自动密钥加密法中功能较弱的版本冠以 Vigenere 之名。这样的后果是，这种较低级的加密法，由于来历不明，即使是在其基本弱点被暴露后的很长时间内也未能被破解。然而，Vigenere 加密法成了历史上最著名的多码加密法之一，今天仍旧把它作为多码加密法的一个基本范例。

Vigenere 加密法是基于关键词的加密系统，但不是像单码关键词加法那样使用关键词来定义替换模式的，关键词写在明文的上面，并不断重复书写，这样每个明文字母都与一个关键词的字母关联。例如，如果关键词为“hold”，而明文为“this is the plaintext”，那么，关键词—明文的关联如下所示：

```
h o l d h o l d h o l d h o l d h o
t h i s i s t h e p l a i n t e x t
```

每个明文字母与关键词的一个字母配对。例如，第一个明文字母“t”与关键词的字母“h”配对。而明文倒数第 4 个字母“t”与关键词的字母“l”配对。利用如图 3.1 所示的 Vigenere 表，这些字母对就可用来确定明文字符的加密结果。这种加密法的操作与 Alberti 的加密转盘差别不大。用密钥字母确定表的行，而明文字母确定表的列，表中行列交叉处的字母就是用来替代明文字母的密文字母。例如，上面示例的第一对是“ht”。在 Vigenere 表中查找“t”列和“h”行，找到结果密文字母“a”。重复这个过程，可以生成以下的密文：

密钥	h	o	l	d	h	o	l	d	h	o	l	d	h	o	l	d	h	o
明文	t	h	i	s	i	s	t	h	e	p	l	a	i	n	t	e	x	t
密文	a	v	t	v	p	g	e	k	l	d	w	d	p	b	e	h	d	h

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
b	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a
c	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b
d	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
e	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
f	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
g	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f
h	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g
i	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h
j	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i
k	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j
l	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k
m	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l
n	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m
o	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
p	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
q	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
r	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
s	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
t	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
u	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
v	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
w	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
x	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
y	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
z	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y

图 3.1　Vigenere 表

注意，密文“v”出现了两次，一次是替代明文“h”，另一次是替代明文“s”。这个示例就体现了多码加密法的基本性质，即，同一个密文字母可以用来表示多个明文字母。

要破解 Vigenere 加密法，需颠倒该查找过程。利用写在明文上面的密钥和密钥-密文对，在 Vigenere 表中确定相应的明文。要实现这些，找到由密钥确定的行，扫描该行，直到找到密文字母，该密文字母所在列的第一个字母就是明文字母。

当然，CAP 软件提供了 Vigenere 加密法。像前面一样，先输入明文码，选择 Vigenere 加密法，输入一个密钥，其余的 CAP 软件将帮助你完成。

3.1.1 Vigenere 加密法分析

Vigenere 加密法强大的功能持续了好几百年。它抵制住了所有破解它的尝试，这主要归功于它不可破解的传奇，但后来最终还是破灭了。该加密法之所以能维持这么长的时间，一部分原因是，破解单码加密法的标准密码分析工具对这种多码加密法背道而驰。由于密文中字母的出现频率都差不多，使用频率高的字母不见了。你可以想像 Bob 和 Alice 使用 Vigenere 加密法的喜悦心情，只要有了这种加密法，就可以立即用来相互发送消息。例如，Eve 截获了以下的密文：

```
hptns bzjke mgmtc hlxrc wtbjx htapi wnqrl ecnqv oykga esixi mlvck eobqw
tzxgz eqzqq rpifm nrinp ofzkq pzzve neugw slogw ivvqa wpuch edwoi rpinq
idbco edqpx hpxcw tmcvr ohqvl iysyi acmqr tzxqj tsmrv omtgq fpmnj rpmvs
spvfq elvax htviw iykgm tywym sdmey rpiie iyavl eckqr seipx aebcg kdqns
ovnqa aclvs hpitm nrnts mjwww ozvdc tsmye yswys femph ojwwx htvma edpqy
lokje nrmqy raiuw wzzfw iykgx htaem psmte pamcv sewdi spkwv exiaf ehmee
nowkx oykge mzvvl lpboi kywya hlbas ucbjs urpvw oybjm stauy elzg
```

由于不知道加密法的改变，Eve 仍然使用其正常的工具来破解，即进行频率统计。但令她很奇怪的是，其结果是她从未见过的。所期待出现的高峰和低谷不见了，如图 3.2 所示。

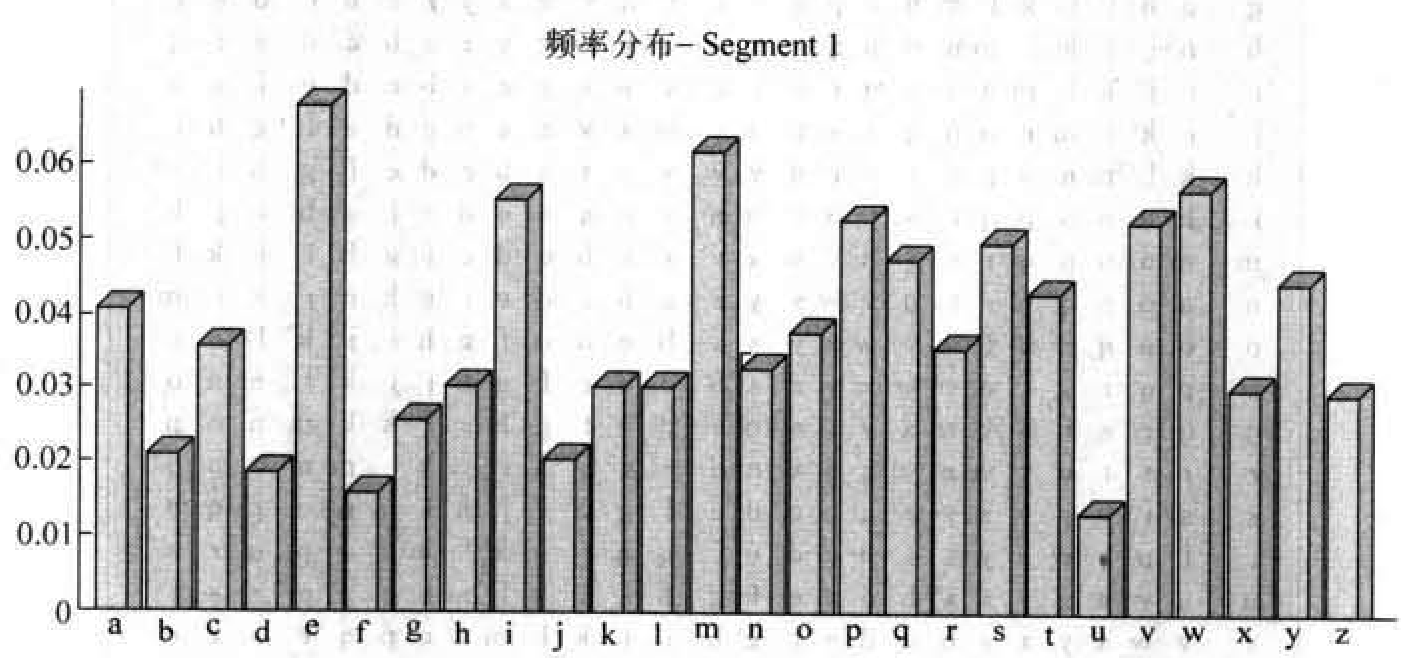


图 3.2 Eve 的频率分布图

由于怀疑该加密法不再是单码加密法（当频率分布比期望的更平滑时，这是一个很好的猜想），Eve 猜测 Alice 和 Bob 可能换成了多码加密法。她通过使用一个测试（称为**一致性**

索引(Index of Coincidence, IC))证实了她的猜想。

频率分布的平滑暗示该加密法为多码加密法,IC 是用来区分单码加密法和多码加密法的另一个工具。IC 是 1922 年由 William Friedman(可能是美国最伟大的密码学家)开发的,它是最早使用数学统计能力的秘密分析工具之一。

IC 是基于**凹凸度量**(measure of roughness, MR,它表示了字母分布频率的高低)理念的。例如,字母完全平滑分布生成的是一种毫无意义的频率分布,如图 3.3 所示。在频率分布图中,凹凸分布则更像标准英语的频率分布,有高峰也有低谷。给凹凸度赋予一个数值的关键是,从密文中随机选取一个字母,看看其出现的概率。如果密文出现的概率分布完全平滑,那么选取的每个密文字母的出现频率都是相同的。因此,也就是说,如果选取"a",其概率为 1/26;选取"y",其概率也为 1/26。然而,由于标准英语字母的频率分布不是完全平滑的,随机选取"a"的概率比随机选取"y"的概率大。因此,凹凸度量为从文本中选取字母的实际概率与从完全平滑分布中选取字母的概率之差。设 p_α 为从密文中选取字母 α 的概率,那么从完全平滑的文本中选取 α 的概率偏差为:

$$(p_\alpha - 1/26)^2$$

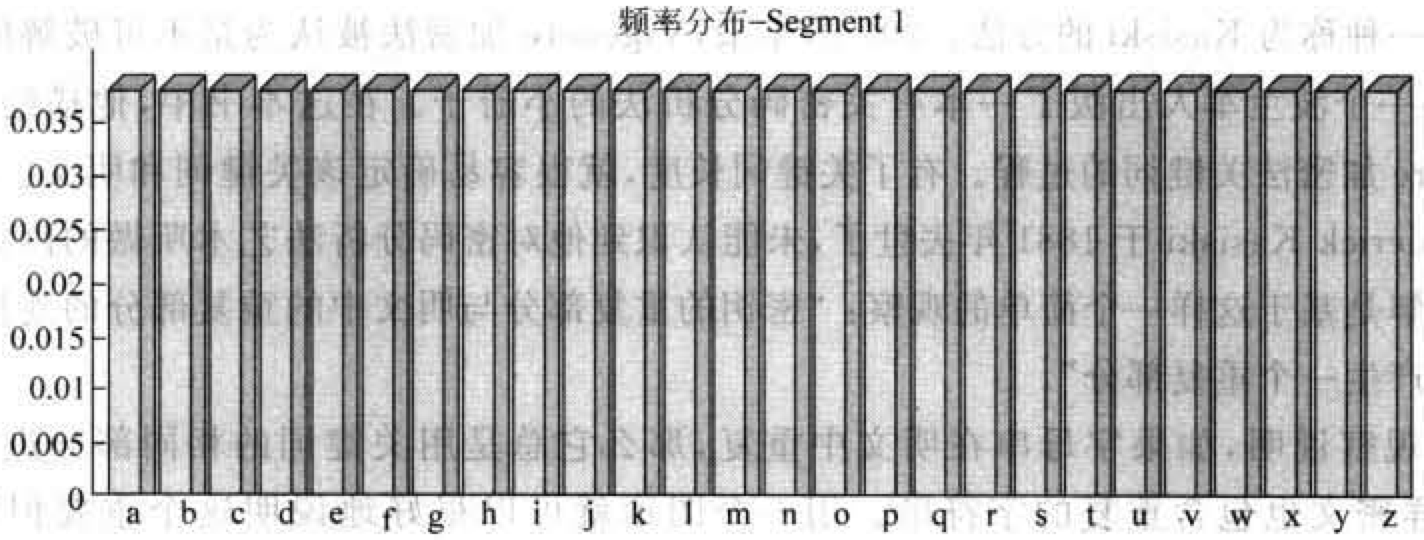


图 3.3　平滑的频率分布图

将差取平方是为了使其值总为正。MR 是完全偏差,因此从 $a \sim z$ 所有字母的偏差和为:

$$\mathrm{MR} = \sum_{\alpha=a}^{z}\left(p_\alpha - \frac{1}{26}\right)^2$$

展开上式,并进行一定的取舍计算后得:

$$\mathrm{MP} = \sum_{\alpha=a}^{z} p_\alpha^2 - 0.038$$

对于某个密文,如果能估算出 p_α^2,那么 MR 就能确定。由于 p_α 为从密文中选取字母 α 的概率,那么 p_α^2 就是从全部密文中选取字母 α 的概率与从剩余密文(即第一个 α 字母去除后的密文)中选取字母 α 的概率的倍数。如果密文有 n 个字母,f_α 是密文中 α 字母的个数,那么:

$$p_\alpha = f_\alpha / n$$

因此

$$p_\alpha^2 = (f_\alpha/n)[(f_\alpha - 1)/(n-1)]$$

MR 中依赖于密文的惟一之处是 p_α^2 的和,因此,IC 可以定义为:

$$\mathrm{IC} = \frac{\sum_{\alpha=a}^{z} f_\alpha(f_\alpha - 1)}{N(N-1)}$$

单码加密法的 IC 值大概为 0.066。对于完全平滑文字，其值为 0.038，如果 IC 值位于 0.038 与 0.066 之间，那么该密文的加密法可能是多码加密法。事实上，利用下表，IC 能暗示出多码加密法的密钥长度：

密钥长度	IC 值	密钥长度	IC 值
1	0.0660	7	0.0420
2	0.0520	8	0.0415
3	0.0473	9	0.0411
4	0.0450	10	0.0408
5	0.0436	11	0.0405
6	0.0427	12	0.0403

Eve 从 CAP 软件中就截取的密文运行 IC 测试，得 IC 值为 0.041，这清楚地表明，Alice 和 Bob 已经改换成多码加密法了，且其关键词长度为 5～10。利用这些事实，Eve 决定找些能破解这个最著名的多码加密法（即 Vigenere 加密法）的工具。

好了，用不了多久，Eve 就可以找到很多可以破解 Vigenere 加密法的工具。最著名的工具是一种称为 Kasiski 的方法。300 多年来，Vigenere 加密法被认为是不可破解的，直到 1863 年一个波兰军人出版了一本有关密码分析法的小册子。在这本书中，他描绘了确定 Vigenere 加密法关键词的过程。有了关键词长度，就很容易确定该关键词和明文。遗憾的是，Frederick Kasiski 于 1881 年去世了，未能认识到他对密码分析法艺术所做的巨大贡献。他的破解是基于这样一个简单的观察："密钥的重复部分与明文中的重复部分的连接，在密文中也产生一个重复部分"。

该观察说明，如果字母串在明文中重复，那么它总是用关键词的相同部分进行加密的，这样密文也包含重复的字符串。用一个图示就可以很好地说明这个重要但简单的思想：

```
R U N R U N R U N R U N R U N R U N R U N R U N R U N
t o b e o r n o t t o b e t h a t i s t h e q u e s t
K I O V I E E I G K I O V N U R N V J N U V K H V M G
↑←——— 9 个字符 ———→↑        ↑←—— 6 个字符 ——→↑
```

在这种情况下，关键词为"RUN"。注意，重复的明文字母"tobe"是用关键词字符串"RUNR"加密的。这将生成重复密文字符串"KIOV"。在这个示例中，明文字母"th"也是重复的，由相同的关键词部分"UN"加密，生成相同的密文字符串"NU"。该观察令人感兴趣的不是密文字符本身，而是密文中重复字符串的间距，密文重复字符之间的间距反映了密钥重复使用的次数。这反过来又为发现密钥长度提供了非常重要的信息。正如后面将介绍的那样，这是这些类型的加密法的致命弱点，因为，知道了密钥的长度，就能够确定实际的密钥内容。

例如，关键词"RUN"的长度为 3。注意到重复密文字符串"KIOV"的间距为 9。重复密文字符串"NU"的间距为 6。这些间距都是关键词长度的倍数（6＝2×3，9＝3×3）。这并不奇怪，因为，为了使关键词本身与重复的明文对齐，它必须自己重复整数次。因此，下面是

Kasiski 建议的破解过程：

(1) 找到密文中重复的字符部分；

(2) 计算重复字符之间的字符数；

(3) 找出从步骤(2)中得到的数的因子；

(4) 最大公约数很可能就是关键词的长度。

密文中可能会出现"accidental"。这是一种巧合，将给确定关键词长度提供错误的线索。但是，这种偶然性重复的影响可以削弱，而真正的重复是会集中在某个数上的。因此，最大公约数(或因子)是关键词长度的最好猜测。

一旦确定了关键词的长度，余下的问题就只是如何使用该信息去找到真正的关键词了。解决这个问题要求你想想，关键词的长度真正意味着什么。通常，在加密分析法中，有关密文的每个信息都是具有一定暗示的。这些暗示有些被证明为是非常有用的，有些则没用，但都值得去考虑。在这种情况，关键词的长度揭示了密文可以被如何破解成单码加密的一个集合。

例如，如果 Eve 发现 Alice 和 Bod 的关键词为 6 个字母长，那么，她就知道，相隔 6 个字母都是用相同关键词字母加密的，如图 3.4 所示。

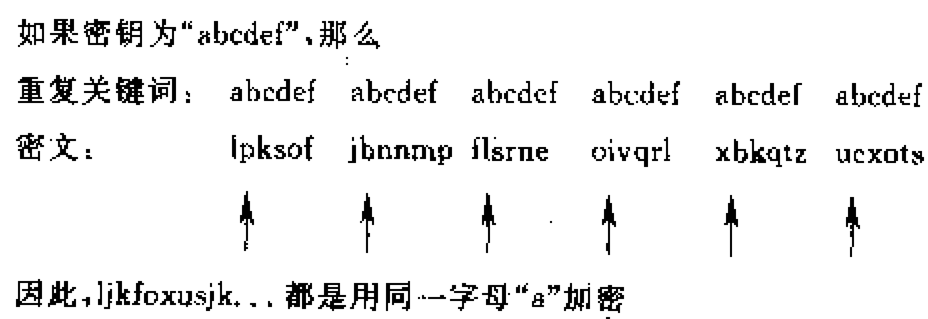


图 3.4 探究关键词的长度

因此，破解 Vigenere 加密法的问题(这里知道了关键词的长度为 n)就变成了破解 n 个不同单码加密的问题。只要密文足够多，可以生成合理的统计样本，单码加密法就可以很容易解决，Vigenere 加密法也就统一解决。

3.1.2 用 CAP 分析 Vigenere 加密法

Eve 现在还是需要破解从 Alice 和 Bob 那里截获来的 Vigenere 密文。但是，利用 CAP 软件，她只有输入密文，然后运行 Kasiski 分析即可。所运行的结果如图 3.5 所示。CAP 软件建议最后的关键词长度为 5，这与 IC 测试一致。如果关键词的长度真为 5，那么相隔 5 个字符将用相同的关键词字母加密。Eve 利用这个信息将密文分为 5 个字母集合。第一集合由第 1、6、11、…个密文字符组成，第二个集合由 2、7、12、…个密文字符组成，等等。每个集合就是一个单码加密密码，因此，Eve 只有运用标准的加密分析法就可以了。

当她尝试破解第一个集合时，她发现仍需要做大量的工作。这是因为她再也找不到单词了，而且构成这个集合的明文字母不是单词，没有任何意义。

对这个问题进行一些研究后，Eve 发现有另外一个更容易的办法来解决从 Vigenere 加

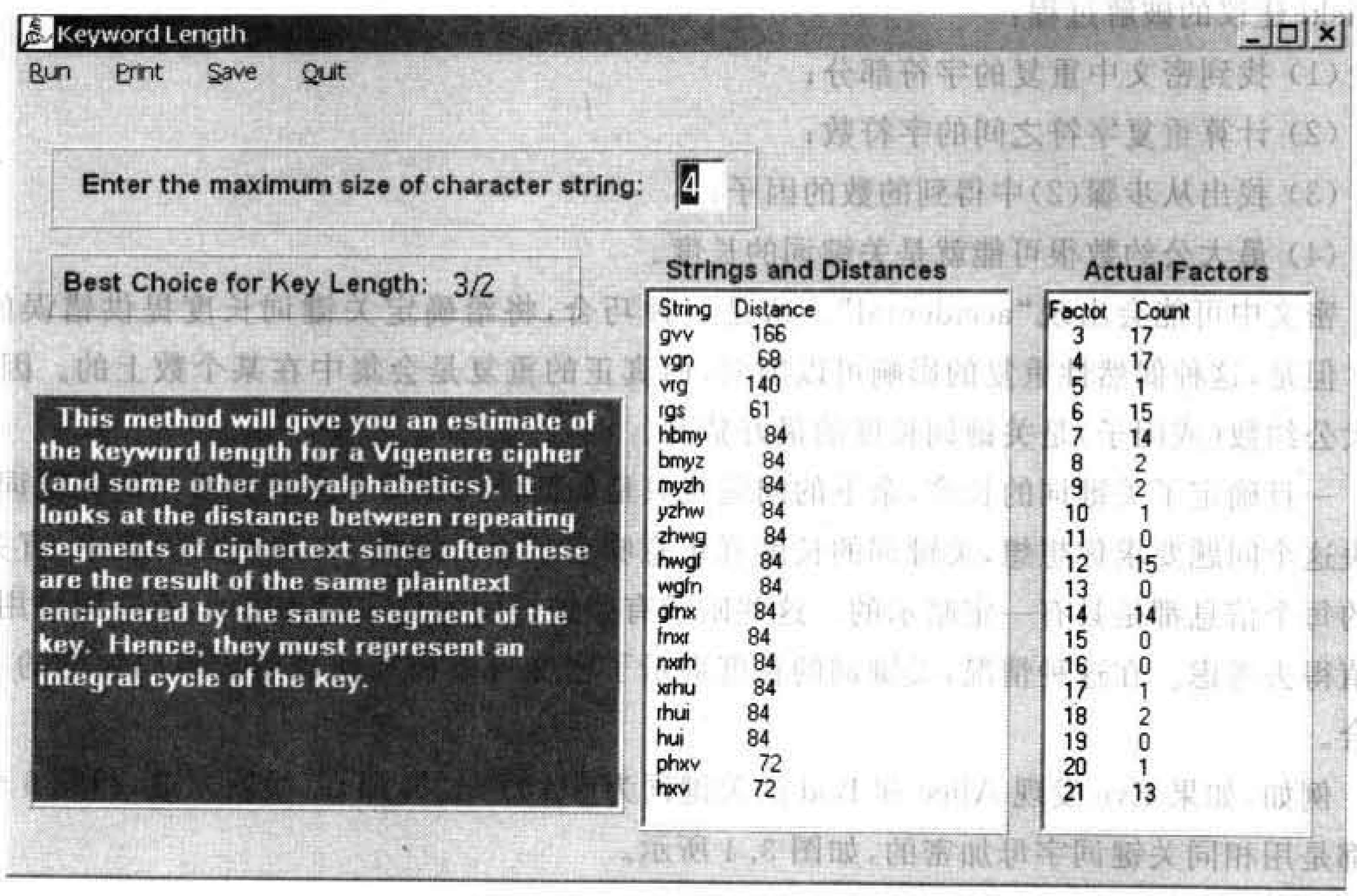


图 3.5 Kasiski 分析法的运行结果

密法中形成的单码子密码。这种方法称为**低频率分析法**(low-frequency analysis)。其过程是,先将目标子密码填入一列中。通过将第 1 列的字母按字母表顺序移动一位形成第 2 列;再将第 2 列的字母同样按字母表顺序移动一位形成第 3 列;重复这样做,直到生成 25 列。这些列表示了原始密文的所有可能移位情况,其中之一就是实际的明文。由于这些列是从原始明文中选取的文字,不可能去扫描列来查找词(甚至是可能的词)。但是,可以利用英语的另一个特征来确定这 25 列中的哪一列包含了真正的明文,这就是标准英语中的低频率字母(如 j、k、q、x、z)的出现频率和约为 2%。在这些列中,如果这 5 个字母的出现频率和大于 2%,就不太可能是正确的明文列。通过去除这些列,最后可能就剩下一列或少数的几列了,从而就容易找到明文列了。

当然,Eve 很乐意尝试这一新技术。首先,她将所截获的密文按上面的方法分成 5 个字母集合,具体如下:

集合 1 hbmhwhweoemeternopnsiweriehtoiattofrsehitsriesakoahnmotyfohenrwihppsseenomlkhuuose

集合 2 pzglttncyslozqprfzelvpdpddpmhyczsmpppltyydpyceedvcprjzssejtdorazytsaepxhoyzpylcrytl

集合 3 tjmxbaqnkivbxziizzuovuwibqxcqsmxmtmmvvvkwmiakibqnlinwvmwmwvpkMizkammwkimwkvbwbbpbaz

集合 4 nktrjprqgxcqgqfnkvggqconcpcvvyqqrgnvfaigyeivqpcnqvttwdyypwmqjQufgetcdwaekgvoyajvjug

集合 5 seccxilvaikwzqmpqewwahiqoxwrlirjvqjsqxwmmyelrxgsasmswceshxayeywwxmevivfexeliasswmy

根据对第 1 个字母集合进行标准的单码分析,Eve 确定关键词的第 1 个字母必定是字母"a"。当她尝试基于第 2 列使用低频率分析法时,她得到了结果的前 3 行(低频率字母的数量标在字符串的末尾)。

Shift 0：

pzglttncyslozqprfzelvpdpddpmhyczsmppplt yydpyceedvcprjzssejtdorazytsaepxhoyzpylcrytl 10

Shift 1：

qahmuuodztmparqsgafmwqeqeeqnizdatnqqqmuzzeqzdffewdqskattdkuepsbazutbfqyipzaqzmdszum 23

Shift 2：

rbinvvpeaunqbsrthbgnxrfrffrojaebuorrrnvaafraeggfxertlbuuelvfqtcbavucgrzjqabranetavn 6

由于字母集合含有 83 个字母，低频率字母 j、k、q、x、z 的出现频率和为 1.66。前 3 行能充分地得出一两个字母。实际上，这表明最适合的移位数字是 15。该移位数表明关键词的第 2 个字母为“l”因为另一个移位数 11 能将密文还原为明文。至此，Eve 的最佳猜测是关键词为“al...”。这时你认为真正的关键词可能是什么？好了，Eve 决定对剩余的字母集合继续进行分析，以关键词为“alice”进行尝试。结果证明这是对的！（这也说明了为什么关键词最好不用包含名字或任何与我们有关的东西。）

当然，CAP 软件提供了实行 Kasiski 和低频率分析的工具。如果 Eve 将密文输入 CAP 软件中，以关键词长度为 5 运行低频率分析，CAP 软件将自动选择正确的关键词，如图 3.6 所示。

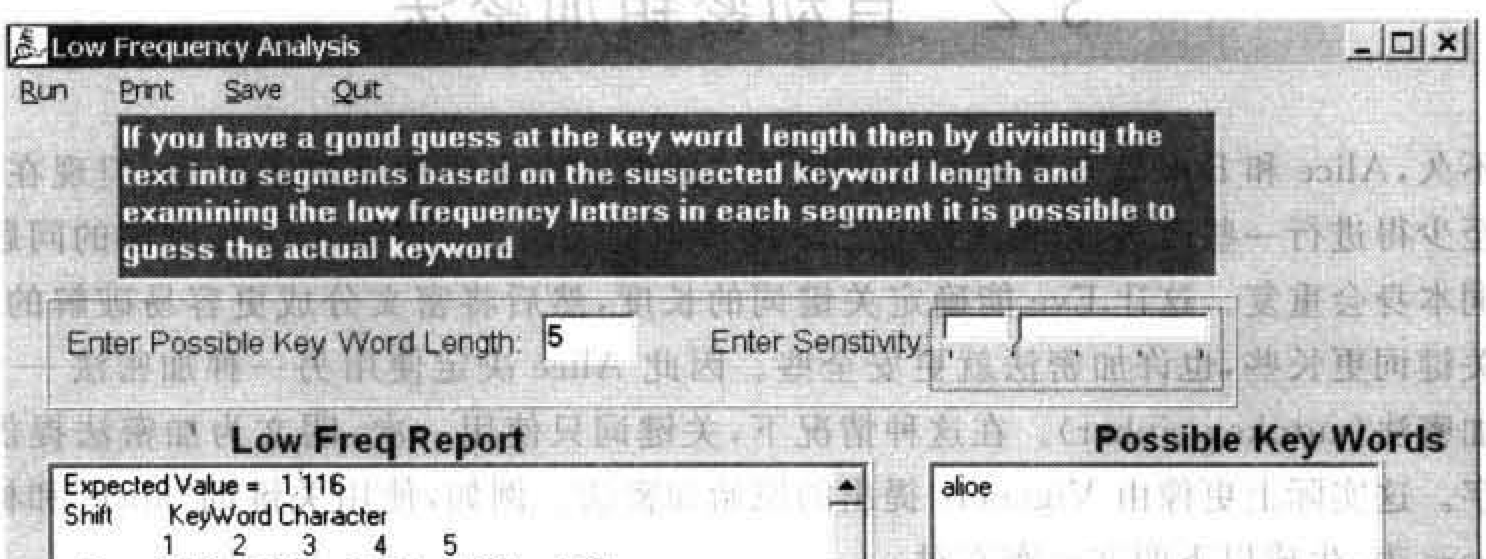


图 3.6　CAP 的 Low Frequency Analysis 运行结果

标准 Vigenere 表不是实现这种加密法的惟一途径。事实上，还有好几种其他的方法。最常见的是 Beaufort 加密法，它使用一个 27×27 的字母表正方形，标准字母表在四条边上重复出现。有一种变异的 Beaufort 加密法，它使用的是 Vigenere 表，但读该表的顺序颠倒了。在变异的 Beaufort 加密法中，密钥字母标示的是行，而明文字母标示的是列。因此，要进行加密，先找到密钥字母行，顺着行找到明文字母，然后往上找到密文字母。其他的变异

加密法有，用关键词设置表（而不是用标准字母表）。例如，如果表的关键词为“cipher”，那么这种 Vigenere 表的第 1 行就是：

c i p h e r a b d f g j k l m n o q s t u v w x y z

其他行通过滚动第 1 行来得到。因此，第 2 行就是：

i p h e r a b d f g j k l m n o q s t u v w x y z c

实践证明，没有哪个变异的加密法比原始的更安全，破解标准 Vigenere 加密法的相同工具和分析过程，同样能很好地破解这些变异的加密法。

令人惊奇的是，人们希望该加密法能去除单码加密法弱点，但实际上它更容易被破解。但是，这并不意味着该加密法就没有使用的价值了。尽管知道了破解过程，该加密法仍使用了一段时间。在美国国内战争时期，南方军就使用这种加密法。更糟糕的是，在整个战争期间，他们只使用三个密钥：Manchester bluff、complete victory 和 come retribution。第一次世界大战前美国同样使用了这种加密法。

3.2 自动密钥加密法

不久，Alice 和 Bob 就知道他们又失败了。他们喜欢多码加密法的思想，但现在认为，他们至少得进行一些改进，以便能防止 Eve 阅读他们的消息。Vigenere 加密法的问题在于关键词本身会重复。这让 Eve 能确定关键词的长度，然后将密文分成更容易破解的片断。如果关键词更长些，也许加密法就更安全些。因此 Alice 决定使用另一种加密法——自动密钥加密法（autokey cipher）。在这种情况下，关键词只使用一次，明文为加密法提供其余的文字。这实际上更像由 Vigenere 提出的原始加密法。例如，使用关键词“alice”和标准的 Vigenere 表，生成以下明文一密文对：

a l i c e i h o p e t h i s w o r k s b e t t e
i h o p e t h i s w o r k s b e t t e r t h a n ...
i s w r i b o w h a h y s k x s k d w s x a t r

这并不是什么新想法。1550 年，物理学家和数学家 Girolamo Cardano 描述了初始的自动密钥加密法，该方法看上去比简单的 Vigenere 加密法更安全。至少，看不出它具有密钥重复的弱点。当然，CAP 软件也同样实现了这种多码加密法，只要从 Ciphertext 菜单中选取 Autokey 项即可。令人惊奇的是，这种加密法并不像它看上去的那样安全，它只是一种具有“运动”密钥的加密法而已。我们将在介绍流加密法的有关章节中遇到这个概念。

这种加密过程的一个变异是使用密文（而不是明文）来提供重复的密钥。此时，前面介绍的消息将显示如下：

a l i c e i s w r i b z f j e p q p b f t j i f
i h o p e t h i s w o r k s b e t t e r t h a n...
i s w r i b z f j e p q p b f t j i f w m q i s

自动密钥加密法看上去像是 Vigenere 加密法一个很好的修改，其实它有一个重大的不

利之处。加密和解密过程中任何地方的一个错误都将影响其后的所有文字。例如，如果 Alice 发送一个消息给 Bob 时偶尔记录了错误的密文字母，那么，当 Bob 解密这一消息时，从错误之处到消息末尾的所有内容都是不可读的。我们到目前为止所介绍的其他加密法就不会有这种问题。在这些加密法中，如果记录了一个错误的密文字母，只有相应的明文字母是错误的。由于只有一个不正确的字母，所以明文仍然保持有很好的可读性。

3.2.1　自动密钥加密法的分析

Alice 和 Bob 同意使用基于密文的自动密钥加密法，也就是说，密文成为密钥（正如他们后面会发现的那样，这是一个糟糕的选择）。Eve 仍然能截获 Alice 与 Bob 通信的消息，并保存了以下一些自动密钥加密的密文：

tzjqf	bzvwq	bctkk	gqnxn	ukgxo	iezqv	madqf	zevin	mspzc
dwkhq	xompf	esdim	mkrvq	erfpb	hgqjh	bvjqh	ucxbl	unfdp

由于 Eve 最初不知道加密法改变了，她仍尝试使用标准的 Vigenere 分析工具。当然，由于这不是 Vigenere 加密法，她无法还原明文。假设 Alice 和 Bob 改变了加密法，Eve 试图确定他们所选取的新加密算法。密文的频率分布仍然表明这是多码加密法（因为频率没有显示单码加密法的高峰和低谷）。但是，关键词长度可能很长，用 Kasiski 方法无法检测。这些向 Eve 暗示 Alice 和 Bob 使用的是字符密钥加密法。

做了这种假设后，Eve 首先尝试最简单的方法。她猜测该密文被用作为运动的密钥。因此她决定探究该加密法最明显的弱点，也就是说，由于前面的密文也是后面密文组的密钥，惟一的问题就是确定原始密钥的结束点以及密文作为密钥的开始点。这是一种基本的强行攻击方法。首先，以移动一位的密文作为密钥，将密文加密。如果未能生成可读的明文，那么以移动两位的密文作为密钥，重复上面的过程，然后是三个字母，等等，直到明文出现。当然，这意味着密文的开始部分没有正确解码。如果有必要，这通过匹配模式词可以很容易完成。

正如前面的一样，CAP 软件为这种加密分析提供了一个自动工具。Eve 运行 CAP 软件后的结果如图 3.7 所示。

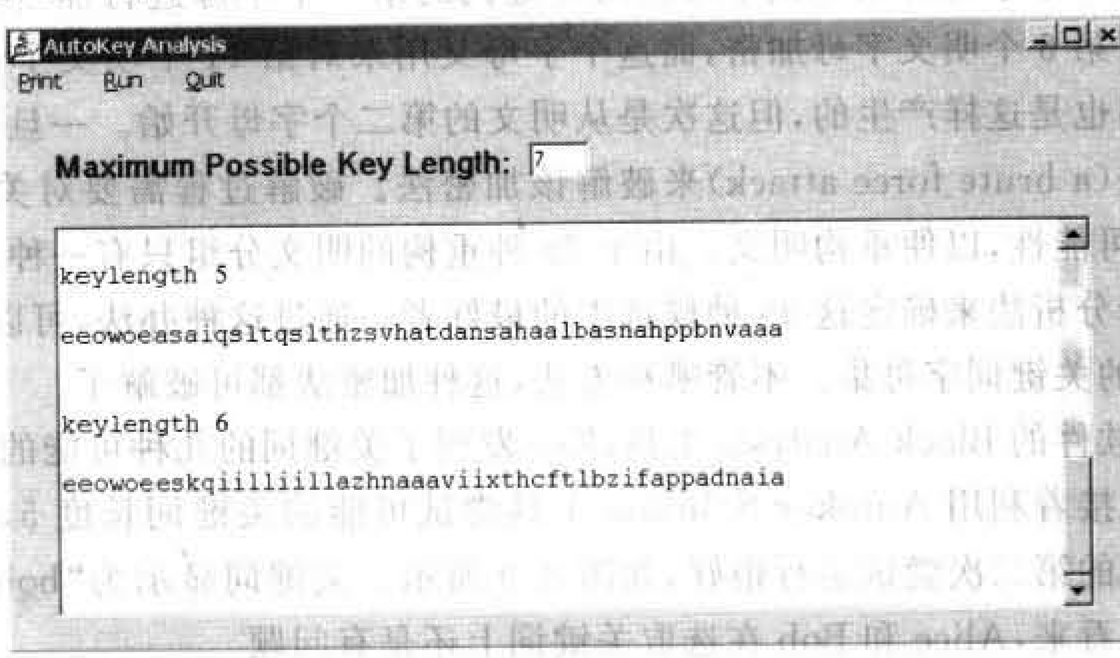


图 3.7　CAP 的 Autokey Analysis 窗口

该加密分析暗示，关键词长度为 5，在头 5 个字母的后面提供了所有的明文。Eve 猜测 Alice 和 Bob 又是选用相同的关键词，并尝试使用 alice，成功了。

Alice 和 Bob 很快发现他们的错误，并认为他们真该使用明文作为运动的密钥。此时，这去除了自动密钥加密法的密文为密钥的弱点，但有引入了新的弱点。

Eve 再次截获了如下一些以新的自动密钥加密法加密的消息：

ucbew cpgqy pbynp mljmg kmuqw wjckg cldjy ktavj hxspp
vvzwj wpsml vxmnv msumn jmikv txygg nabma jkbcm lvvwo gvy

在尝试了自动密文密钥方法后，Eve 确定 Alice 和 Bob 又进行了改变。Eve 怀疑他们是用明文(而不是密文)作为运动的密钥。她尝试关键词“alice”，由于该关键词行不通，Eve 知道 Alice 和 Bob 进行了更好的保护措施。

但从 Eve 的角度来说，并不是无计可施了。有不少可用来破解自动明文密钥加密法的工具，特别是对基于的 Vigenere 加密法。第一个破解工具是用来估计关键词的长度。当然，用来破解 Vigenere 关键词长度的 Kasiski 法对自动明文密钥加密法无效，但是，仔细观察这类加密法的本质，可以发现另一种用来确定关键词长度的方法。

尽管在明文样本加密中，没有定义关键词长度的重复周期，但确实定义了“组长度”，它是密文与将其用作密钥字母来加密其他明文字母之间的距离。例如，如果关键词为“alice”，那么，其组长度为 5。如下所示，第一个明文字母由“a”加密，随后的第 5 个字母(即字母“t”)用来加密明文字母“e”。

|←————→|
a l i c e ⓣ h i s a e x
ⓣ h i s a e x a m p l e

在上面示例中，“h”被加密，然后，其后第 5 个字母用来加密“x”。通常，每个字母都是用其左边第 5 个字母来加密，而每个字母则将其右边第 5 个字母加密。有关破解关键词长度的知识同样适用于组长度。

正如标准的 Vigenere 加密法一样，知道了关键词长度，就可以将密文分成几个分组，然后再对各个分组进行解密分析。如果知道关键词长度为 5，那么每隔 5 个字母是一个分组。第一个分组从第一个字母开始。该字母利用关键词的第一个字母进行加密。然后，第一个明文字母用来对第 6 个明文字母加密，而这个字母又用来对第 11 个明文字母加密，以此类推。第二个分组也是这样产生的，但这次是从明文的第二个字母开始。一旦分组知道了，可以用蛮力攻击法(a brute force attack)来破解该加密法。破解过程需要对关键词的第一个字母尝试 26 种可能性，以便重构明文。由于 26 种重构的明文分组只有一种是正确的，可以使用低频率字母分析法来确定这 26 种候选中的最好者。通过这种办法，可以立即确定关键词，或者是可能的关键词字母集。不管哪种方法，这种加密法都可破解了。

使用 CAP 软件的 Block Analysis 工具，Eve 发现了关键词的几种可能的长度，如图 3.8 所示的输出。她接着利用 Autokey Solution 工具尝试可能的关键词长度 5，但没有生成合理的关键词。她的第二次尝试运行很好，如图 3.9 所示。关键词显示为“bob”，利用它就可以解密密文了。看来，Alice 和 Bob 在选取关键词上还是有问题。

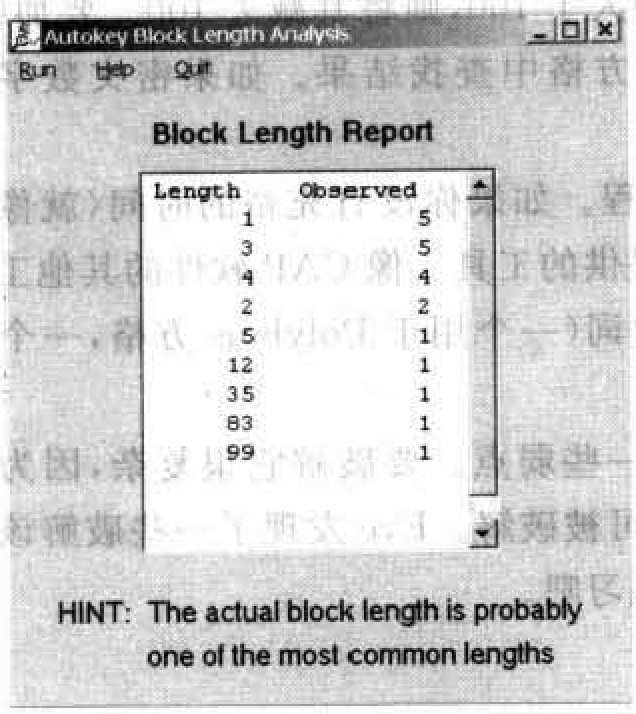


图 3.8　Autokey Block Length Analysis 窗口

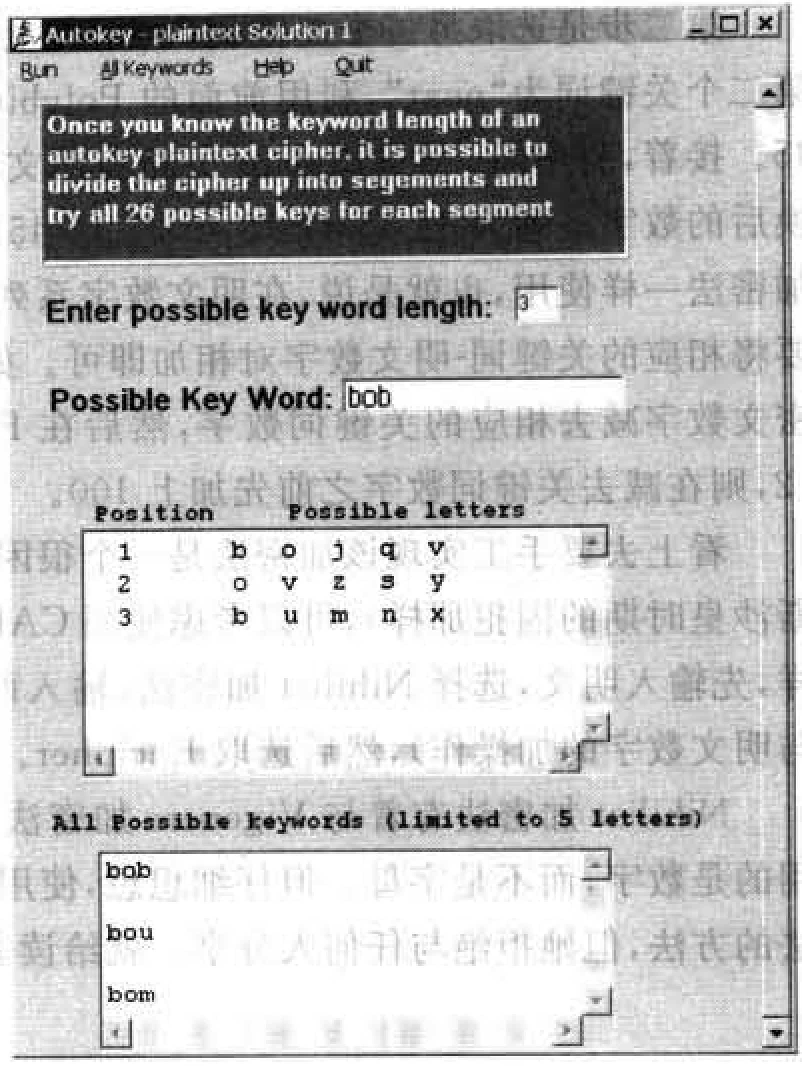


图 3.9　Autokey Solution 窗口

3.3　Nihilist 加密法

由于 Alice 和 Bob 使用的多码加密法只有一种转换类型，没有取得多大的成功，因而决定采用另一种加密法，即 Nihilist 加密法。这种加密法看起来比他们以前使用过的加密法都要安全，因为该加密法包含了不止一种加密操作。

Nihilist 加密法起源于沙皇时期俄罗斯地下党所使用过的一种加密方法。它用于囚犯之间的通信。那时候，大多数俄罗斯囚犯被单独监禁，禁止与其他囚犯通信联系。最初，囚犯们是敲打墙壁，敲打的次数表示棋盘上的字母位置。他们使用这种加密法非常熟练，能以每分钟 10～15 个单词进行通信。

这种以棋盘加密的方法最终发展成 Nihilist 加密法的核心。Nihilist 加密法的第一步是选取一个关键词以构成 Polybius 方格，也就是说，在 5×5 的矩阵中填写该关键词（去除重复的字母），然后继续在其中按顺序填写字母表中的其余字母（删去字母“j”，这样方格中就只有 25 个字母）。用数字 1～5 给方格的行和列加编号。假设关键词为“example”，那么，Polybius 方格为如下所示：

	1	2	3	4	5
1	e	x	a	m	p
2	l	b	c	d	f
3	g	h	i	k	n
4	o	q	r	s	t
5	u	v	w	y	z

第二步是选取另一个关键词，利用以上的 Polybius 方格将其转换成数字。例如，如果第二个关键词为“next”，利用前面的 Polybius 方格，则该关键词转变成数字系列 35 11 12 45。接着，利用同一个 Polybius 方格将明文转换成数字。如果明文为“stop that”，那么，转换后的数字系列为 44 45 41 15 45 32 13 45。从这点来看，Nihilist 加密法可以像 Vigenere 加密法一样使用，也就是说，在明文数字系列下重复写出关键词数字系列。要生成密文，只要将相应的关键词-明文数字对相加即可。如果其和大于 100，则将其减去 100。要加密，用密文数字减去相应的关键词数字，然后在 Polybius 方格中查找结果。如果密文数字小于 12，则在减去关键词数字之前先加上 100。

看上去要手工实现该加密法是一个很困难的过程。如果你没有充裕的时间（就像俄罗斯沙皇时期的囚犯那样），可以考虑使用 CAP 软件提供的工具。像 CAP 软件的其他工具一样，先输入明文，选择 Nihilist 加密法，输入两个关键词（一个用于 Polybius 方格，一个用于与明文数字的加操作），然后选取 Encipher。

Nihilist 加密法有着与 Vigenere 加密法同样的一些弱点。要破解它很复杂，因为它使用的是数字，而不是字母。但仔细想想，使用数字也可被破解。Eve 发现了一些破解该加密法的方法，但她拒绝与任何人分享。就给读者留作练习吧。

3.4 圆柱面加密法

放弃使用 Nihilist 加密法后，Alice 和 Bob 转向了另一种多码加密法，该加密法是基于一个圆柱面的。第 1 章介绍过轮（或圆柱面加密法）的概念，它是基于托马斯·杰斐逊发明的加密设备的。由 Etienne Bazeries 于 1891 年发明的圆柱面加密法可以算得上是最著名的圆柱面加密法之一了。Bazeries 是一个很有才华且成功的加密分析员。他很擅长破解加密法，但不善于发明加密法。他只是发明了一种圆柱面加密法，现在该加密法就是以他的名字来命名的，即 Bazeries 圆柱面加密法。1891 年，他试图让法国军队采纳这种加密法，但失败了。但是，30 年后的 1922 年，美国军队使用了他的加密法。

Bazeries 圆柱面加密法由 20 个轮组成，每个轮上的字母表顺序不同，如图 3.10 所示。这些轮按预先选定的顺序排列。（这个顺序就是该加密法的密钥。）转动这些轮，使明文出现在同一条直线上，然后可以选取任意的其他直线上的字母作为密文。例如，如果只使用 3 个轮，其顺序为 1 5 3，明文为“hat”，那么，“sxh”就可以作为一个密文，如图 3.11 所示。

在现在的计算机年代，Alice 和 Bob 再也不用造 20 个金属轮了，他们可以使用 CAP 软件提供的 Bazeries 圆柱面加密法工具，就可以进行加密和解密。像以前一样，在 CAP 软件中输入明文，从 Cipher 菜单选取 Bazeries Cipher 选项，将弹出如图 3.12 所示的窗口。密钥就是这 20 个轮的顺序。CAP 软件的默认顺序是 1，2，…，20。当然，Alice 和 Bob 不会使用这个默认值，因此输入一个不同的轮顺序，例如，他们的密钥为 2，5，10，20，1，3，19，17，18，15，4，9，11，16，6，7，14，13，8，12。有了这个密钥后，他们继续进行相互通信，认为他们终于打败了 Eve。

1	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	x	y	z
2	b	c	d	f	g	h	j	k	l	m	n	p	q	r	s	t	v	x	z	a	e	i	o	u	y
3	a	e	b	c	d	f	g	h	i	o	j	k	l	m	n	p	u	y	q	r	s	t	v	x	z
4	z	y	x	v	u	t	s	r	q	p	o	n	m	l	k	j	i	h	g	f	e	d	c	b	a
5	y	u	z	x	v	t	s	r	o	i	q	p	n	m	l	k	e	a	j	h	g	f	d	c	b
6	z	x	v	t	s	r	q	p	n	m	l	k	j	h	g	f	d	c	b	y	u	o	i	e	a
7	a	l	o	n	s	e	f	t	d	p	r	i	j	u	g	v	b	c	h	k	m	q	x	y	z
8	b	i	e	n	h	u	r	x	l	s	p	a	v	d	t	o	y	m	c	f	g	j	k	q	z
9	c	h	a	r	y	b	d	e	t	s	l	f	g	i	j	k	m	n	o	p	q	u	v	x	z
10	d	i	e	u	p	r	o	t	g	l	a	f	n	c	b	h	j	k	m	q	s	v	x	y	z
11	e	v	i	t	z	l	s	c	o	u	r	a	n	d	b	f	g	h	j	k	m	p	q	x	y
12	f	o	r	m	e	z	l	s	a	i	c	u	x	b	d	g	h	j	k	n	p	q	t	v	y
13	g	l	o	i	r	e	m	t	d	n	s	a	u	x	b	c	f	h	j	k	p	q	v	y	z
14	h	o	n	e	u	r	t	p	a	i	b	c	d	f	g	j	k	l	m	q	s	v	x	y	z
15	i	n	s	t	r	u	e	z	l	a	j	b	c	d	f	g	h	k	m	o	p	q	v	x	y
16	j	a	i	m	e	l	o	g	n	f	r	t	h	u	b	c	d	k	p	q	s	v	x	y	z
17	k	y	r	i	e	l	s	o	n	a	b	c	d	f	g	h	j	m	p	q	t	u	v	x	z
18	l	h	o	m	e	p	r	s	t	d	i	u	a	b	c	f	g	j	k	n	q	v	x	y	z
19	m	o	n	t	e	z	a	c	h	v	l	b	d	f	g	i	j	k	p	q	r	s	u	x	y
20	n	o	u	s	t	e	l	a	c	f	b	d	g	h	i	j	k	m	p	q	r	v	x	y	z

图 3.10 Bazeries 圆柱面加密法的轮

1	5	3
h	a	t
i	j	v
j	h	x
k	g	z
l	f	a
m	d	e
n	c	b
o	b	c
p	y	d
q	u	f
r	z	g
s	x	h
t	v	i
u	t	o
v	s	j
x	r	k
y	o	l
z	i	m
a	q	n
b	p	p
c	n	u
d	m	y
e	l	q
f	k	r
g	e	s

图 3.11 用 Bazeries 圆柱面加密法加密明文"hat"

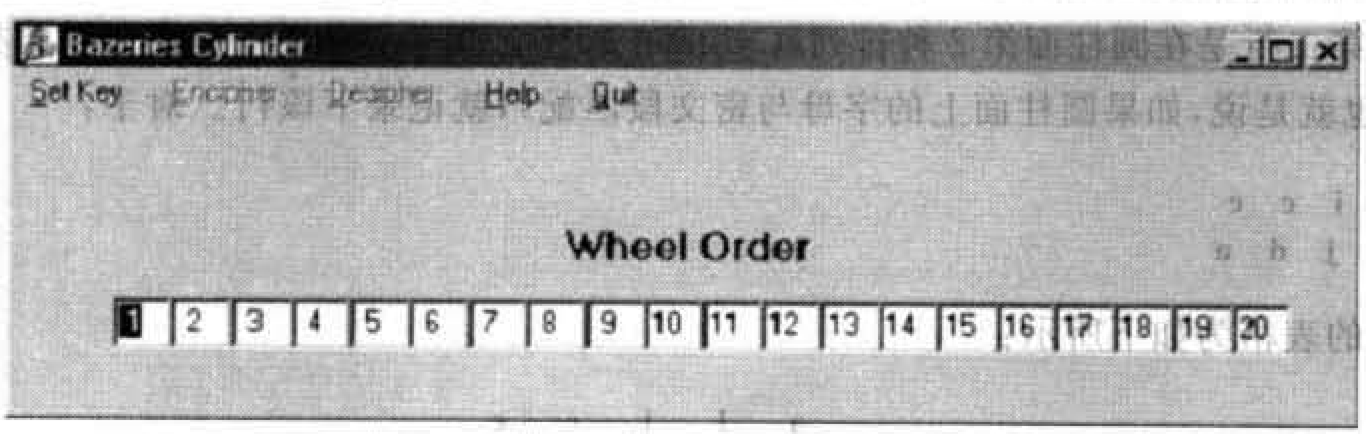


图 3.12 Bazeries Cylinder 窗口

需要注意的很重要一点是,CAP 软件为 Bazeries 圆柱面加密法添加了一个特征。在 CAP 软件中,用户无须去选择用圆柱面上哪条线的字母作为密文,CAP 会不断改变所选取的直线。可能选取紧邻明文的直线作为第一个块(包含有 20 个密文字母),而隔一行的直线作为第二个块,隔两行的直线作为第三块,等等。在间隔了 24 行后,将重复上面过程。当使用 CAP 的工具来破解该加密法时,记住这一点很重要。

3.4.1 Bazeries 圆柱面加密法的分析

一段时间里,Alice 和 Bob 的通信比较安全,因为 Eve 被新的加密法难倒了。她知道这仍是多码加密法(为什么?),但要破解却有困难。一旦她发现了这是 Bazeries 圆柱面加密法,她就知道如何破解它。事实上,破解该加密法的技巧早在 1893 年就被 Marquis de Viaris 发现了,因此,称之为 de Viaris 法。

该方法假设这样两个事实:第一,该加密法是 Bazeries 圆柱面加密法,且已知圆柱面的

内容；第二，至少知道了一个明文单词。因此，这是已知明文攻击法。其目标是确定圆柱面的实际顺序。

Eve 发现了这种攻击法，决定对以下密文进行尝试：

ekjdahdujcmvjbtfllhjrfgtfwfespbftpnxxgvaxpirabhusltcuuecdzabhgyfrsaahdihhhsjylnn

由于该消息是从 Bob 发送给 Alice 的，Eve 假设单词“alice”出现在其中的某个地方。这样 Eve 将所有需要的信息应用到 de Viaris 攻击法。其过程是，就已知单词扫描密文，查找能既能产生已知单词，又能得到该密文段的圆柱面排列顺序。这里，Eve 从密文的前 5 个字母开始匹配已知单词：

a l i c e
e k j d a h d u j c m v j b t f l l ...

问题是：圆柱面必须这样排列，已知的单词出现在某条直线上，而明文则出现在另一条直线上。如果没有找到这种排列，则将已知单词后移一个字母，重新查找这种排列，如下所示：

** a l i c e**
e k j d a h d u j c m v j b t f l l ...

从已知单词与密文的第一个可能对齐开始，Eve 为所有可能的圆柱面排列创建一个表格。该表格的顶部是已知单词，第一行是在圆柱面第 1 种排列(Cylinder 1)中已知单词行的下一行；第二行是在圆柱面第 2 种排列(Cylinder 2)中已知单词行的下一行；以此类推。如果匹配(也就是说，如果圆柱面上的字母与密文段匹配)，就记录下该行。对于：

a l i c e
e k j d a

Eve 的表格为如下所示：

	a	l	i	c	e
1	b	m	j	d	f
2	e	m	o	d	g
3	e	m	o	d	b
4	z	k	h	b	d
5	j	k	q	b	a
6	z	k	e	b	a
7	l	o	j	h	f
8	v	s	e	f	n
9	r	f	j	h	t
10	f	a	e	b	u
11	n	s	t	o	v
12	i	s	c	n	z
13	u	o	r	f	m
14	i	m	b	d	u
15	j	a	n	d	z
16	i	o	m	d	l
17	b	s	e	d	l
18	b	h	u	f	p
19	c	b	j	h	z
20	c	a	j	f	l

如果每列至少有一个匹配，那么这就可能是已知单词与密文的匹配。如果至少有一列不匹配，那么已知单词就不是用这种密文字母的对齐方式加密的。因此，必须将已知单词在密文字母上移动。这里，Eve 为已知单词的每个字母都找到了一个可能的匹配，因此她尝试查找一个对这 5 个字母都适用的圆柱面排列（如果失败，就需要将已知单词移动一个字母。）。Eve 的表分析显示，第 2 或第 3 种圆柱面排列可能在第一位，因为"e"与"a"正确匹配了；第 4、5 或 6 种圆柱面排列可能在第二位，因为"k"与"l"正确匹配了。其他可能的情况是，第 1、7、9、19 或 20 种圆柱面排列可能在第三位；第 1、2、3、15、16 或 17 种圆柱面排列可能在第四位；第 5 种圆柱面排列可能在第五位。尽管这并不能完全确定圆柱面的顺序，但它大大减少了可能顺序的数量。这里，前 5 个圆柱面可能是如下的顺序之一：

圆柱面位置	1	2	3	4	5
圆柱面编号	2	4	1	1	5
	3	6	7	2	
			9	3	
			19	15	
			20	16	
				17	

例如，已知可能的顺序是 2、4、1、3、5，另一种是 2、4、1、15、5。这总共有大约 60 种可能的圆柱面顺序，Eve 必须对每一种进行尝试。当然，这都是针对密文的前 5 个字母，并产生单词"alice"。要测试它们，Eve 需要使用 20 个密文字母，因为总共有 20 个轮。

接下来，由前 5 个轮加密的密文段组成如下：

ekjda hdujc mvjbt fllhj rfgtf wfesp bftpn xxgva xpira bhusl tcuue cdzab hgyfr saahd ihhhs jylnn

Eve 尝试用圆柱面顺序为 2、4、1、3、5 解密，结果如下：

alice hdujc mvjbt fllhj pherh wfesp bftpn xxgva ssful bhusl tcuue cdzab cktex saahd ihhhs jylnn

这看上去希望不大，但当她尝试使用 2、6、1、3、5 解密，密文变成了：

alice ciphe succe mvjbt fllhj pherh wfesp bftpn xxgva ssful bhusl tcuue cdzab cltex saahd ihhhs ytext

这看起来是一种可能性，第二个单词可能是 cipher，第三个单词可能是 successful，最后一个单词可能是 text。这就可以用圆柱面的匹配模式来解开其他字母了。Eve 最后发现，Alice 和 Bob 使用的圆柱面匹配模式为：2、6、1、3、5、4、7、8、9、10、11、12、13、14、15、16、17、18、19、20。

用手工完成这些工作量很大，因此 CAP 软件提供了一些帮助工具。一个工具是用来查找一个可能的单词匹配。它可显示出已知单词中每个字母所在的圆柱面号，如图 3.13 所示。另一个工具是利用这些可能的圆柱面排列来确定密文中其他的单词，如图 3.14 所示。

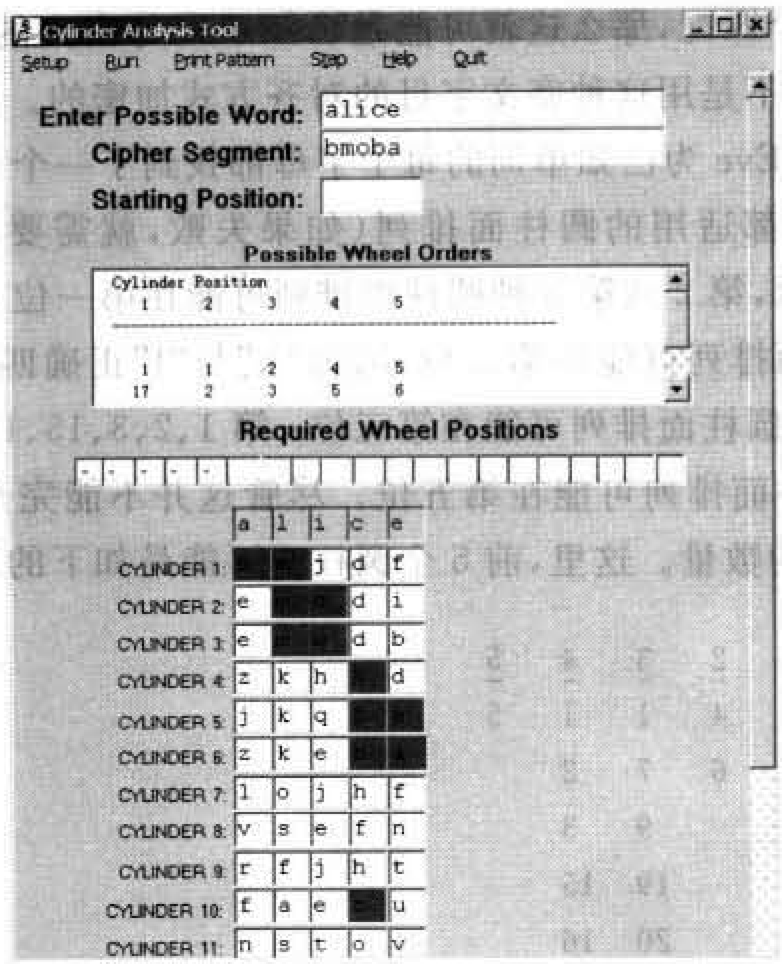


图 3.13　Cylinder Analysis Tool 窗口

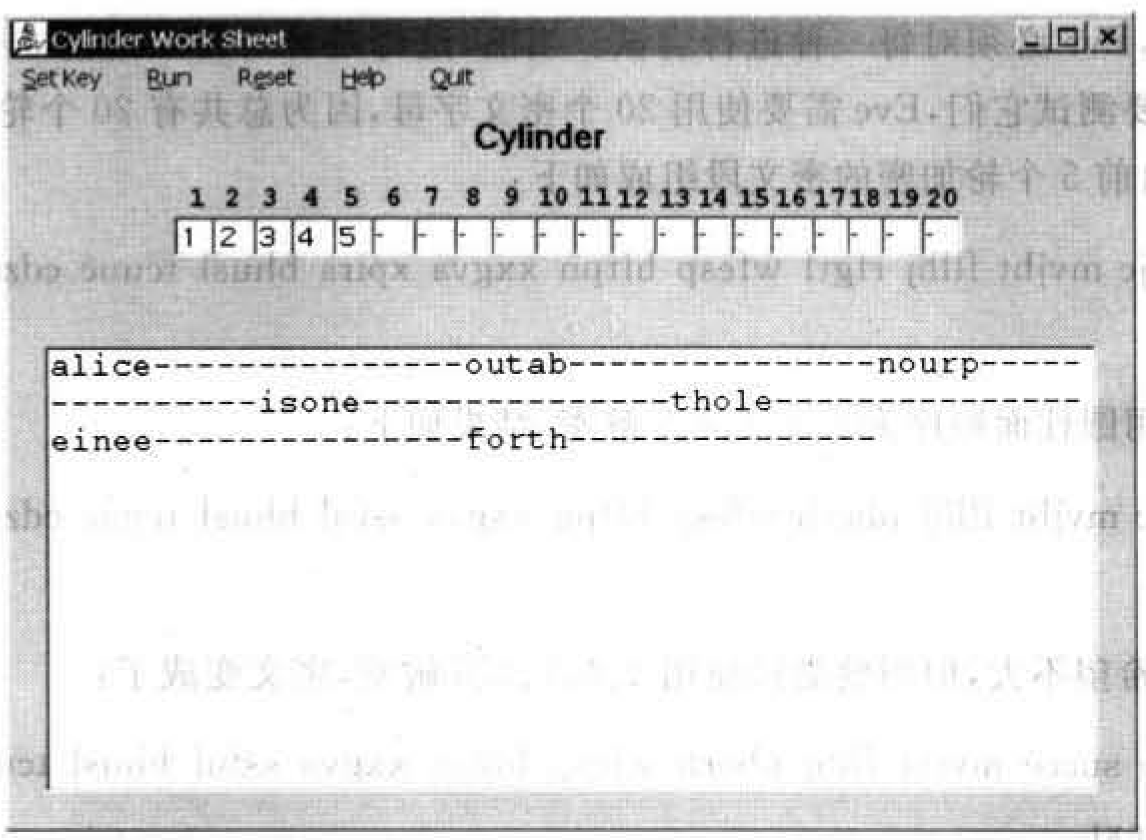


图 3.14　Cylinder Work Sheet 窗口

3.5　回转轮加密法

由于圆柱面加密法最终还是被 Eve 破解了，Alice 和 Bob 认为需要重新考虑整个加密过程。在查看了加密法的历史后，Bob 发现回转轮(rotor)加密法很像二战时期德国使用的那种。他发现这种加密法很有趣，但决定不用它，因为这种加密法在二战时就已经被盟军破

解了。

回转轮加密法是用电机系统来实现多码变换的方法。回转轮是一个圆盘，它的两面都有电子接点，每个接点代表字母表中的一个字母。回转轮内部有连接各接点的电线，这种连接方式定义了单码替换方式。一个简单的回转轮如图 3.15 所示。由于每个回转轮是在内部用电线连接，以表示字母表中 26 个字母的替换模式，因此其替换模式的数量有：

26! =403 291 461 126 605 635 584 000 000

如果将几个回转轮串连起来，并以不同的速率转动，就可以构建成一个功能强大的多码加密系统。

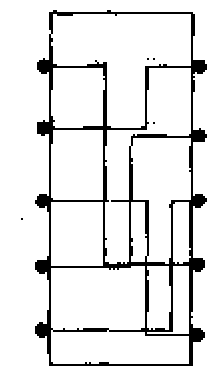


图 3.15　一个简单的回转轮示意图

图 3.16 是一个双轮系统示例。开始时，"A"对应于"B"，但当轮 2 转动 1 位后，"A"就变成了对应于"E"。当轮 2 转动 25 位后，轮 1 的字母"A"转动 1 位。因此，总共有 26×26=676 种可能的替换模式。随着回转轮数量的增加，替换模式数也将大大增加。

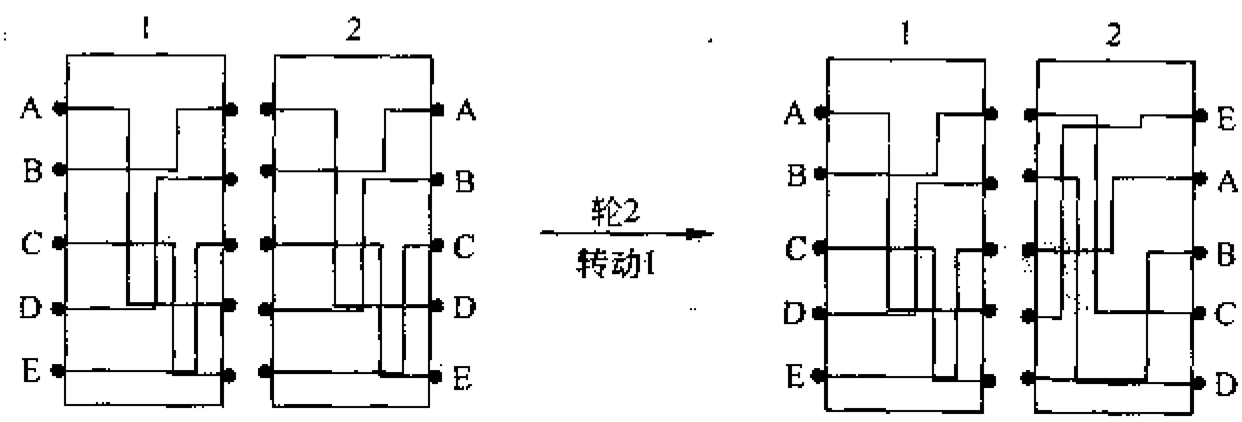


图 3.16　一个双轮系统示例

二战时期，德国使用了这种类型的回转轮系统，称为 Enigma 加密法。令德国意想不到的是，在战争爆发之前，英国就已经破解了 Enigma 加密法。英国率先开始了研究德国的 Enigma 加密法，因为它们掌握了三位杰出的波兰数学家的研究工作。

20 世纪 20 年代，由于害怕德国日益的强大，波兰开始研究德国的加密技术。波兰加密研究局(Polish Cipher Bureau)取得了大量成功研究，可以阅读德国的加密通信，直到 1928 年德国开始使用 Enigma 加密法为止。在德国开始使用 Enigma 加密法之后不久，波兰就破解了该加密法。德国给其在波兰的大使送去了一台 Enigma 加密机。波兰的邮政人员偷偷地先将该机器转交给了波兰有关当局，花费了几天时间研究该机器的构造和操作后，才邮递给德国大使。即使是有了这些重要的信息，还是用了好几年的时间才将该加密法破解。

直到 20 世纪 30 年代早期，才由 Marian Rejewski、Henryk Zygalski 和 Jerzy Rozycki 发现了 Enigma 加密机的一个弱点，并重新构建了一个解密的机器。这可能是密码分析历史上最令人吃惊的成功事件了。他们的研究工作在德国占领波兰之前交给了英国。英国的密码破解员利用波兰的数据破解了德国的加密法。这是关于二战最严格保守的机密之一，战争结束后很长时间内仍没有公开。

德国使用了 Enigma 加密的多种变体，但基本上都是由 3 个回转轮、一个反射器和一个线路连接板组成。实际上，这相当于有 5 个回转轮，其中的 3 个可以选择在任何时候使用。反射器相当于使每个回转轮使用了两次。线路连接板是一个固定的字母替换方式，通常只有一些字母。这个完整的系统如图 3.17 所示，在该图中，"a"对应于"c"。

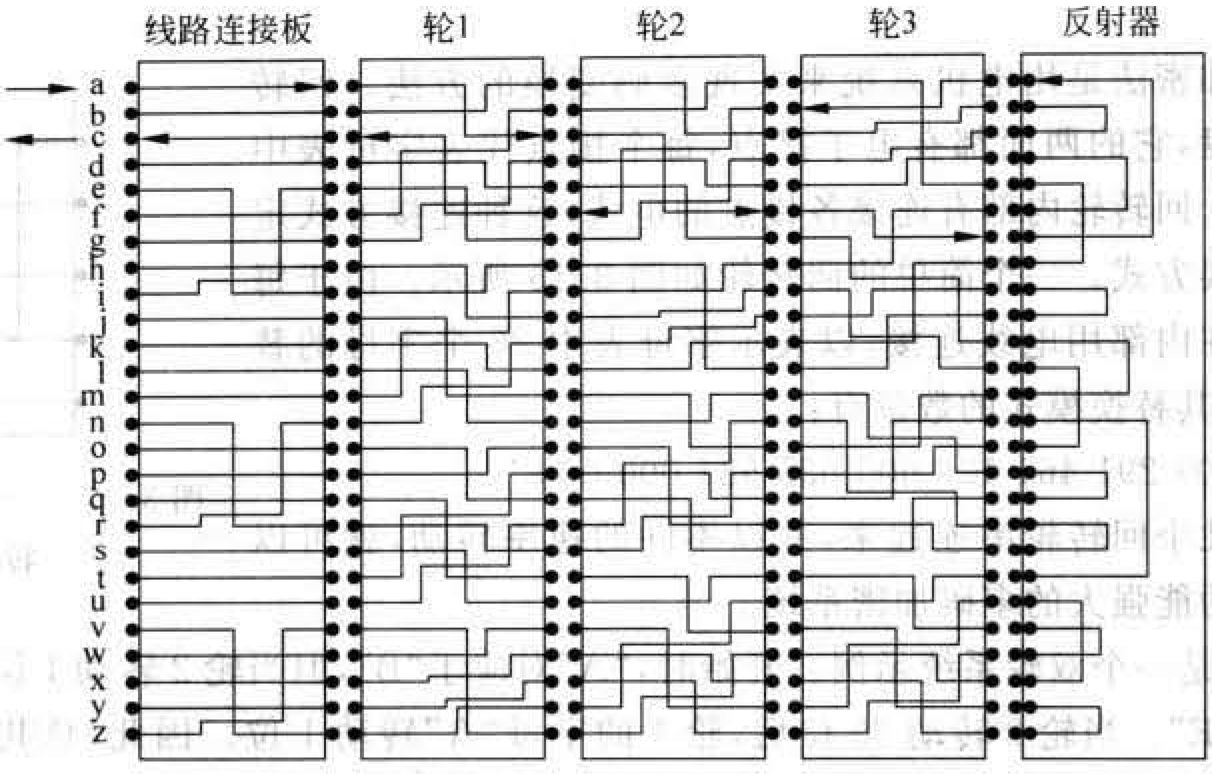


图 3.17　一个有 3 个回转轮的 Enigma 加密系统

CAP 软件提供了一个不带线路连接板的三轮系统。从 Cipher 菜单中选取 Rotor 项，将弹出如图 3.18 所示的窗口。

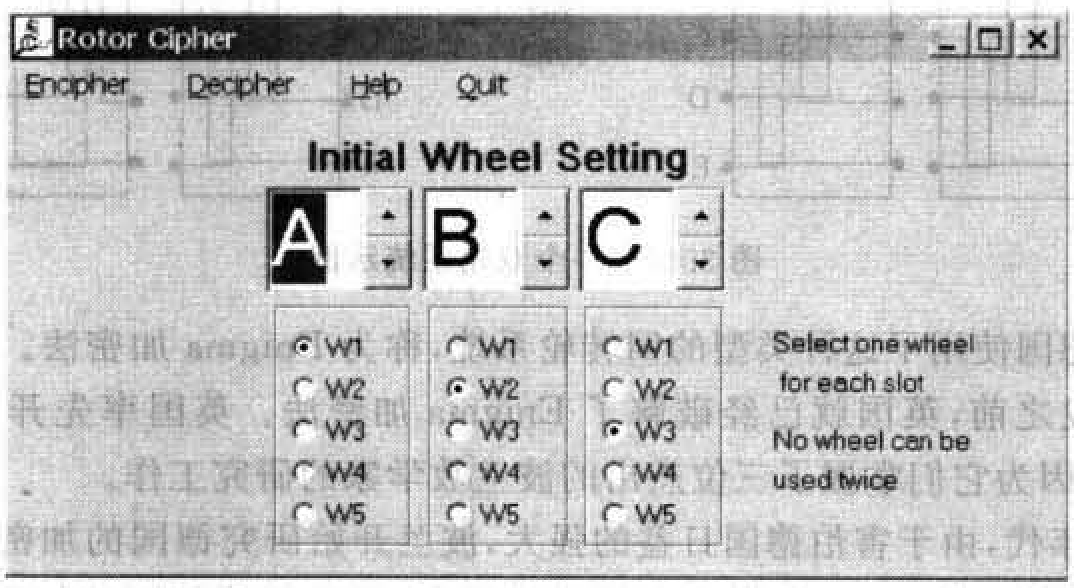


图 3.18　Rotor Cipher 窗口

3.5.1　Enigma 加密法的破解

二战期间，发明了多种攻击 Enigma 加密法的方法和通用回转轮系统。但是，最终找到 Enigma 加密密钥的更容易的方法是，充分利用德国人设置密钥的过程。每个 Enigma 加密操作员从编码簿中选出一个公用的每日密钥，该编码簿含有一个月的密钥。该每日密钥组成了三个回转轮的顺序，每个回转轮的起始位置，以及线路连接板的设置。这个过程有一个严重的缺陷。这样做就意味着，在某一天，德国军队所传送的消息都是用同一个密钥加密的。德国担心如果盟军获得了这么多的消息，可能会发现密钥，从而可以阅读这天的所有消息。这是很现实的担心。这就是为什么即使是在具有如此功能强大的加密法的今天，密钥也要按一定的规律改变。因此，为了弥补这些缺陷，每个消息有它自己的密钥，该密钥是由操作员选择的。当然，操作员同时需要传送这个新的密钥，因为接收员只知道每天的官方密

钥。这就是有关密钥管理的问题。关于这个问题的现代解决方式将在第 9 章介绍。创建一个公共密钥并发送加密消息的步骤如下：

(1) 按官方的每日密钥设置加密机；

(2) 输入操作员为消息选择的密钥(三个字母)；

(3) 重复第(2)步，再次输入操作员所选的密钥；

(4) 将轮还原到由操作员密钥确定的起始位置；

(5) 发送消息。

例如，某天的官方密钥可能是一个特定的线路连接板设置，轮 2、1 和 4，轮的起始设置为“X T B”。操作员可能选择一个随机的起始设置，如“J Y Q”。当 Enigma 加密机设置了每日密钥后，操作员可以输入这个随机设置两次：“J Y Q J Y Q”然后，操作员将轮设置为“J Y Q”，然后传送消息。在接收端，利用官方的每日密钥，接收员会将前 6 个字母解密成“J Y Q J Y Q”，然后将轮的初始位置重新设置为“J Y Q”，以解密该消息的其余部分。将操作员随机选取的密钥发送两次是为了提供解密的可靠性，因为如果这三个字母不重复，接收员可能会认为出了什么错误。但是，这也为盟军破解提供了机会。

有时候，操作员可能会偷懒，反复使用这三个字母，或者选用的是一种简单的模式，例如，三个字母完全相同。使用时间久了，这两种错误都将导致加密法被破解。

但是，这种方法还有一个更重要的问题。它是使用了操作员所选取的密钥两次进行加密的，对经验丰富的加密分析员来说，密文的头 6 个字母就可能会暴露每日密钥的一些信息。例如，如果经 Enigma 加密机加密的信息的头 6 个字母是“RTY UIO”，那么我们就可以知道，操作员的密钥的第一个字母先被每日密钥映射到了“R”，通过 3 个回转轮后又映射到了“U”。

你可能会想：“这只有 6 个字母，谁能从这么少的信息中得到什么？”随着你对密码学知识的不断增加，这个答案可能会让你大吃一惊。知道了头 6 个字母的很多信息后，每日密钥所决定的回转轮的初始位置就可能被确定。

例如，假设盟军在某一天中截获了一些消息。每条消息的头 6 个字母如下：

RTY UIO DEF PCQ NZK WAE PTI SVX SHQ LKG LCM DVY

在这些字母中有几种令人感兴趣的模式。例如，在第一条和最后一条消息中，“R”映射到了“U”，而“U”又映射到了“R”：“RTY UIO UCM RVY”。这形成了一个循环，即“RU”。在其他消息中，“D”映射到了“P”，“P”映射到了“S”，“S”映射到了“L”，而“L”又映射回了“D”。这就形成了一个“DPSL”循环。事实证明，由每日密钥产生的这种循环可以用来确定实际密钥。通过先将所有循环产生的可能密钥归类，一旦通过发现了某个每日密钥的循环结构，就可以从密钥归类中找到每日密钥了。关于 Enigma 的使用方法或规则，其目的是为了更安全，但实际上是适得其反了。

3.5.2 使用 CAP 软件破解回转轮加密法

由于现今计算机的强大功能和速度，完全可以利用强力攻击法来破解回转轮加密法了，如果有了一些已知的明文，就更是如此了。CAP 软件就实现了这种攻击法。从 Polyalphabetic 菜单中选取 Rotor Attack 菜单项，弹出如图 3.19 所示的窗口。输入已知的明文，并单击 Run。然后耐心等待一下，因为该攻击法需要花费一些时间。

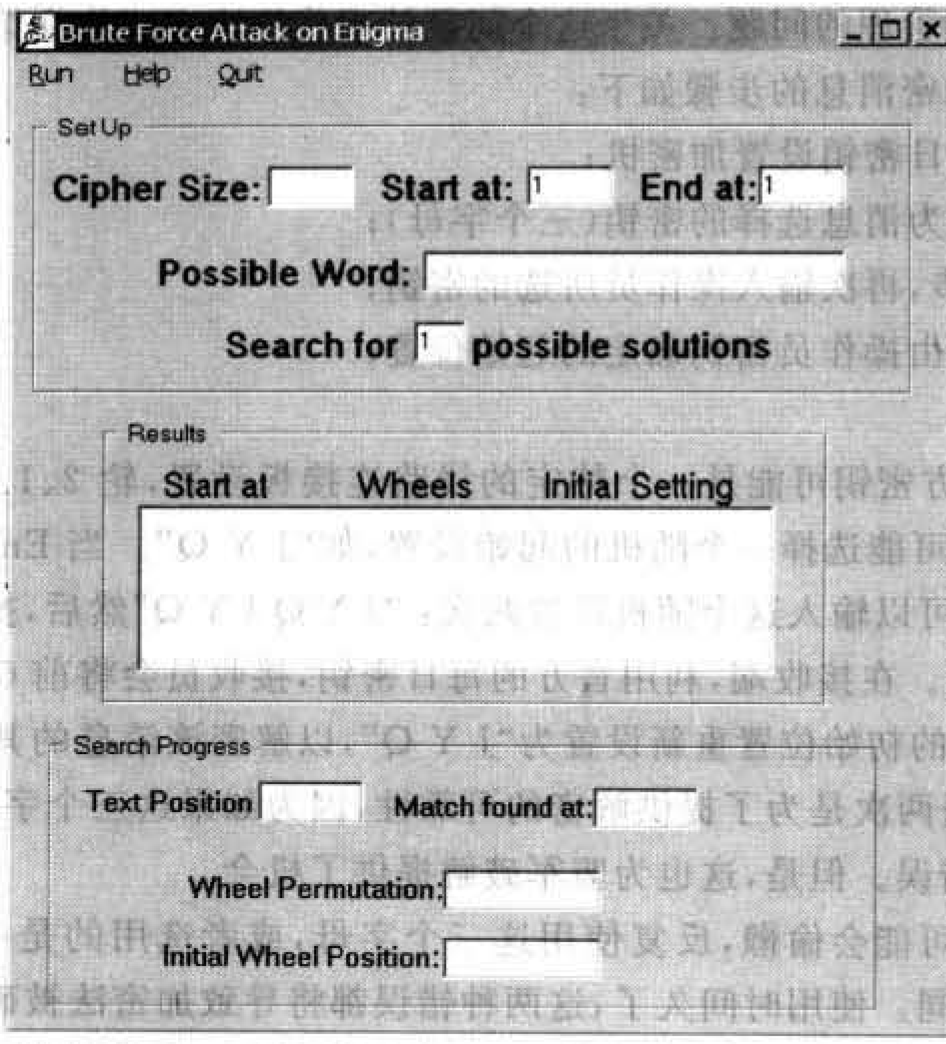


图 3.19 CAP 软件的 Brute Force Attack Enigma 窗口

3.6 加密机的历史简介

19 世纪后期和 20 世纪初，加密机开始应用于明文加密的过程中。加密机的发展过程并不是基于今天的计算机，而是由齿轮组成的基本机械设备，它具有有限的电线、电灯、开关、插销等电子设备。显然，20 世纪 10 年代后期是回转轮机器时代。几乎是在同时，有好几个密码学家各自提出了同样的加密思想。回转轮的概念（最终发展成了 Enigma 加密机）诞生了，并成为了一种新型的多码加密的基础。

Arvid Damm 于 1919 年在瑞典为回转轮加密思想申请了专利，这正是 Boris Hagelin 设计 B-211 加密机的基础。Hagelin 的加密机是在欧洲广泛地使用了几十年的机器之一。当 Damm 和 Hagelin 在欧洲设计回转轮加密机的同时，Edward Hebern 于 1917 年在美国为回转轮加密系统申请了专利。他的回转轮加密机械原理被应用于他的“电子编码机”中，该机器在二战时对美国的加密机影响很大。

当然，最著名的回转轮加密机是 Enigma。盟军能破解该机器，并使用这些信息，从而在整个战争中处于有利位置。一些显然的问题是：美国是如何保护其军事通信的？德国破解了吗？对于上层的通信，美国军队使用的是 ECM Mark Ⅱ（又称为 ECM 或 SIGABA）。与当时大多数的加密系统一样，ECM 也是基于回转轮的加密机。但是，它比 Enigma 复杂得多。它有 15 个回转轮（而 Enigma 只有 3～5 个），并分成三组排列，每组 5 个。每组 5 个回转轮的一面有 26 个接点，代表字母表的 26 个字母。这些回转轮形成一个加密组，如图 3.20 所示，很像 Enigma 加密机，明文字母的信号从回转轮的一面输入，沿一个通道穿过

这 5 个回转轮，从另一面出来就形成了密文信号。Enigma 和 ECM 主要的区别是回转轮的转动方式。在 Enigma 加密机中，第一个回转轮每转动一位就转动一个字母。当转动 26 个位置后，第 2 个轮就转动一位（即转动一个字母），以此类推。而 ECM 回转轮可以在另两组回转轮的控制下以任意方式转动。

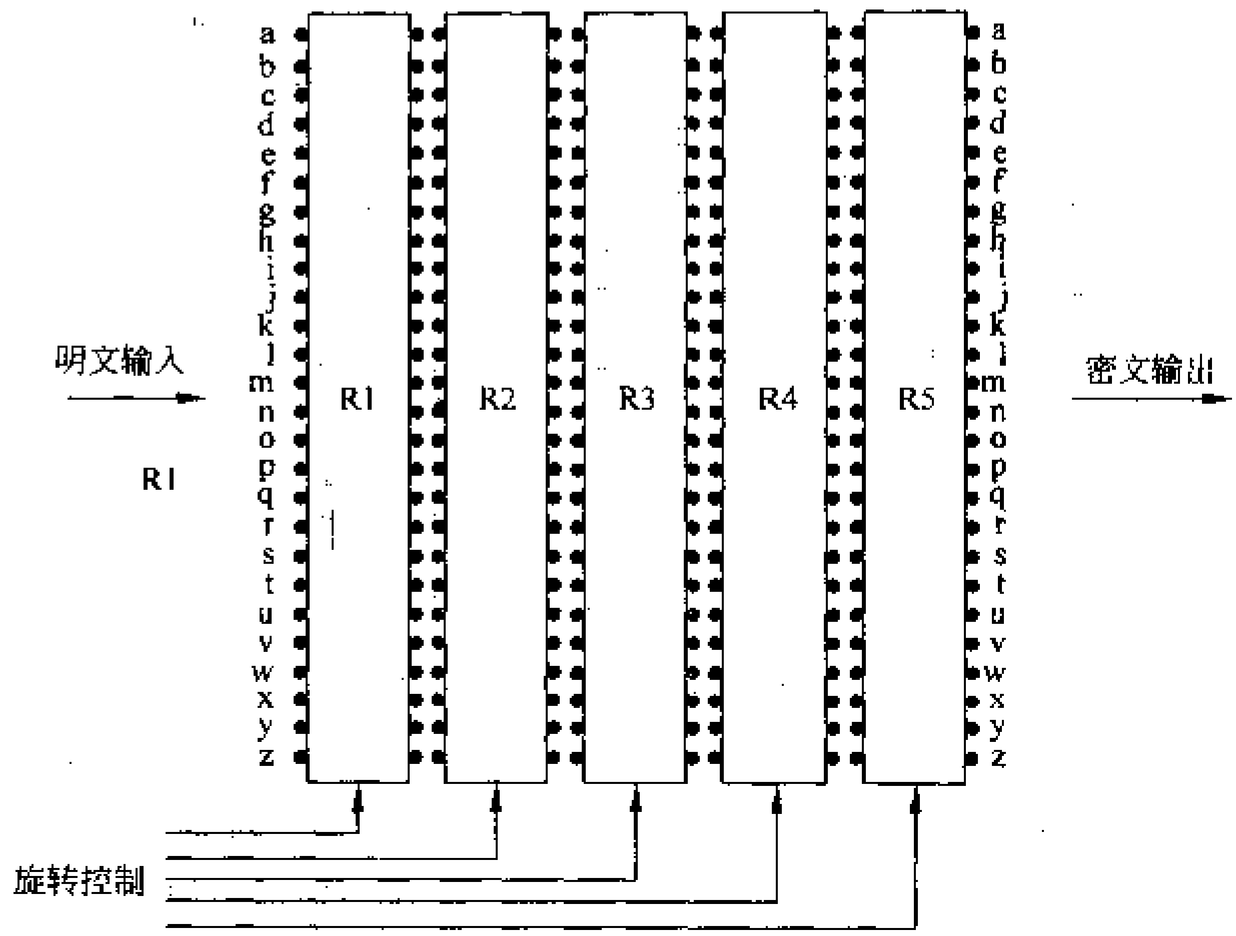


图 3.20　ECM 加密组

控制组也是由 5 个回转轮组成，有 26 个接点，如图 3.21 所示。前 4 个输入固定为"ON"。这些信号可以沿给定的路径通过这 5 个回转轮，并在输出端 1～9 生成"ON"。如图 3.21 所示，一些输出，如输出"k"、"j"和"i"，连接在一起，形成输出 5。这意味着，只要

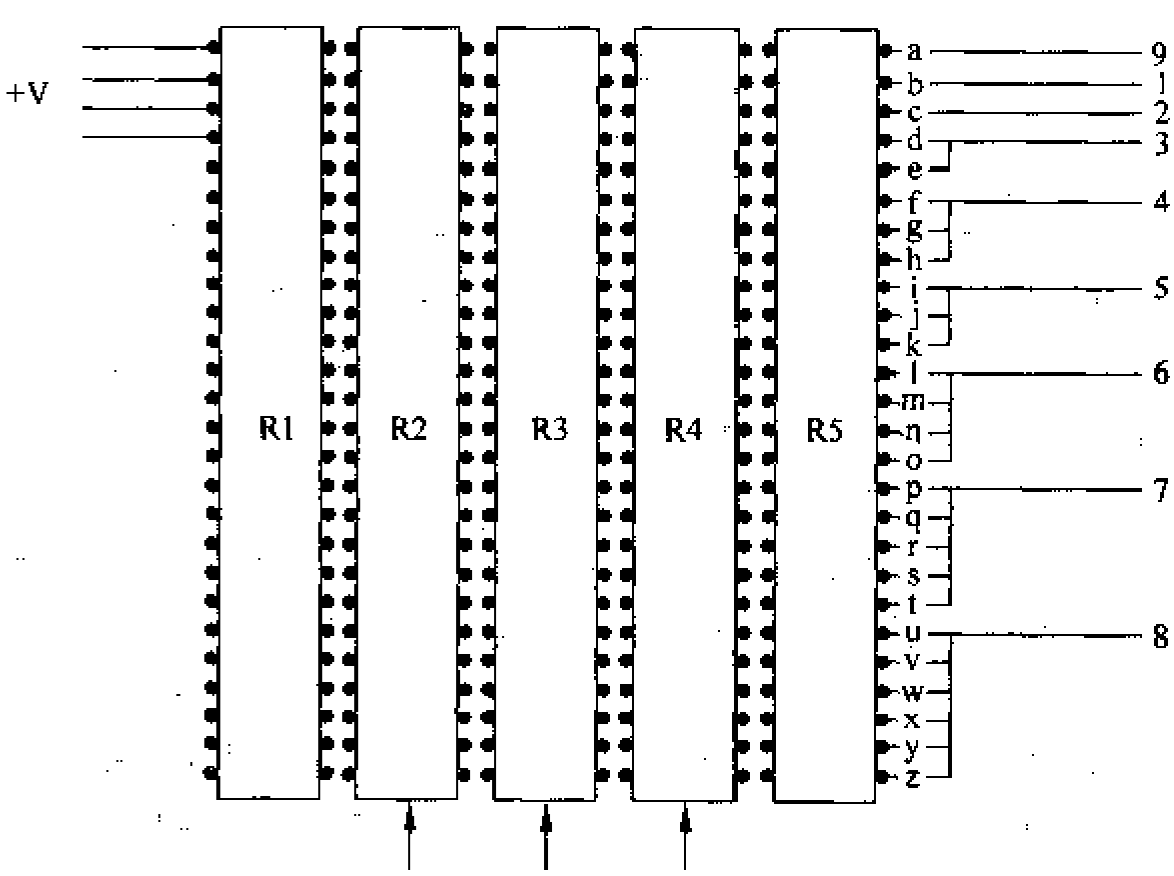


图 3.21　ECM 控制组

“k”、“j”和“i”中有一个或多个为“ON”，那么输出 5 就为“ON”。换句话说，输出是输入的“OR”逻辑运算结果。在控制组中，只有 2、3 和 4 回转轮可以自由转动。轮 3 每转动一位就转动一个字母，转动 26 位后，轮 4 就转动一位，而轮 4 转动 26 位后，轮 2 就转动一位。

控制组发送 10 个输出给索引组。索引组（如图 3.22 所示）的任务是将这 10 个信号转换成 5 个命令集，它们分别对应这 5 个加密回转轮。这些命令决定加密一个明文字母后哪个回转轮将转动。对密码块而言，该结果是随机出现的转动方案。

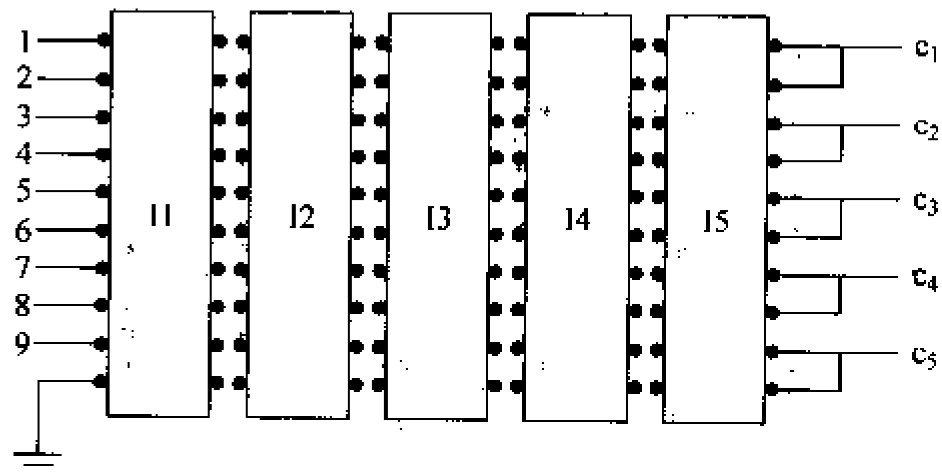


图 3.22　ECM 索引组

二战后的资料表明，尽管联军进行了尝试，但他们从来没有成功破解过 ECM。这部分原因是加密机的复杂度增加了，还有另外的原因是很好地使用了该加密机。在盟军的胜利与联军的失败这两者之间，不仅在于加密法的选用上，还在于管理密钥的方法上。现在请先记住这一点，因为在第 9 章我们将详细介绍它。

直到 1959 年美国海军仍在使用 ECM。ECM 之所以此时才退役，是因为该加密机对新时代的海军来说，其通信速度太慢了。该是改变加密机的本质的时候了。现在的加密机已完全是电子设备的了。这些加密法将在第 6、7 和 8 章中介绍。

3.7 本章小结

与第 2 章中介绍的单码加密法的情况一样，人们也开发和使用了大量的多码加密法。多码加密法都具有这样的特征：一个明文字母可映射到多个密文字母。这种加密方法可以干扰字母出现频率分析法。通常，多码加密法肯定比单码加密法更安全，但还是太脆弱了，任何有一定安全需要的应用都不可使用这种加密法，正如 Alice 和 Bob 所发现的一样。

Alice 和 Bob 与 Eve 之间这个回合的斗争，又是 Eve 获胜了，但她也为获胜付出了很大努力。Alice 和 Bob 在一段时间内取得了优势，但 Eve 最终还是破解了这种加密法。在这个过程中，他们三人都学到了关于加密法及其弱点的更多知识。

Alice 和 Bob 学到了，通过使用多个密文字母来替换一个明文字母的加密法，消平字母的出现频率，从而瓦解简单的频率分析法。他们还知道了，通过使用前面已加密的明文或密文（自动密钥法），生成运动密钥，从而扩展密钥的长度。他们开始使用更加机械化的加密系统，如圆柱面或回转轮加密机。但所有这些在 Eve 的努力下都失败了，他们引入了一些关于加密系统结构的重要思想，这样后面将证明是非常有用的。他们学到的最重要的经验可

能是，加密法看上去复杂的，并不意味着它就是安全的。

Eve 发现，尽管频率分析法有时是一个有用的工具，但并不总是如此。她还知道了查找密文模式的重要性。事实证明，密文的重复部分是破解 Vigenere 加密法的关键。她发现，尽管有功能强大的分析工具帮助，有时要破解某些加密法，还需要很大的耐心，而有时却垂手可得。这种耐心和尝试不同想法直至找到一种有效方法的欲望，将使她可以接受日后更大的挑战。

至此你可能会问，Eve 是怎么总能知道 Alice 和 Bob 改换加密法呢？她是怎么这么快就发现了新的加密法的类型呢？答案是，Eve 可能并不像你想像的这么快。有时，是当她多次尝试仍无法破解 Alice 和 Bob 的消息后，才发现加密法改变了。确定新的加密法的过程，对 Eve 来说有时是漫长而困难的。与所有优秀的加密分析员一样，她需要了解新加密法的本质，查找可能能暴露加密法类型的线索。她还擅长社会工程，这是一个熟练人员的必要技术知识。她可能会向 Alice 或 Bob 询问一些重要信息，但她不知道他们使用的是什么加密法，也不能向他们请求任何建议。她可以从他们的垃圾箱中搜寻有关加密算法的记录。她可能从图书管理员那里获得 Alice 和 Bob 借阅图书的记录，希望其中某本书是加密算法的出处。她可以想尽一切办法，她最终是为了成功。

总之，本章介绍了 5 种新的加密法，具体结构图如图 3.23 所示。

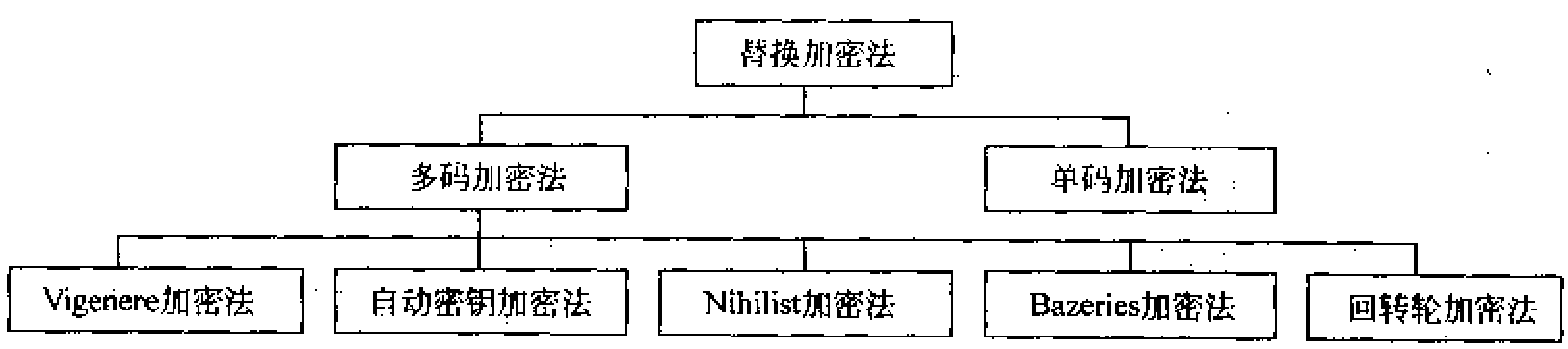


图 3.23 多码加密法

3.8 重要术语

自动密钥法	一种通过把明文或密文作为部分密钥，从而扩展密钥长度的方法。
de Viaris 法	一种已知明文攻击法，通过扫描圆柱面的正确排列顺序来破解圆柱面加密法。
Enigma	二战时期德国使用过的加密法之一。
一致性索引(IC)	由 William Friedman 设计的。它是用来测试多码加密密文的。
同音异义加密法	一种将每个明文字母用多个密文符号来替换的加密法。
Kasiski 法	于 19 世纪由波兰的一个军官发现的，这种方法通过查看重复密文部分，来发现多码密钥的长度。
多码加密法	一种将每个明文字母用多个密文字母来替换的加密法。
回转轮加密法	一种多码加密法，它使用多个回转轮，每个回转轮实现单码加密。这些回转轮可以组合在一起，在每个字母加密后产生一种新的替换模式。

习　题

1. 为 Vigenere 加密法创建一个关键词表，该关键词为“neverhome”，并使用它来加密“alphabet”一词。

2. 请描述一下如何破解 Nihilist 加密法。

3. 利用 Vigenere 加密法，关键词为“section”，将下面一段文字加密成密文，然后进行频率分析，并对结果给以评价。

> Bob, I think Eve may be on to something that will break this cipher. We need to look at the weaknesses that might exist in a Vigenere cipher. If she has broken it, what shall we do? Do you have any idea of where we should turn? Yours, Alice

4. 在问题 3 中所生成的密文的一致性索引是什么，它对发现关键词有何作用？

5. 请填写以下信息：

加密法	操作过程概述	弱点概述
Vigenere 加密法		
自动密钥加密法		
圆柱面加密法		
回转轮加密法		

6. 假设你截获了 Alice 和 Bob 的以下密文：

blakgzomitfraojuilwviifsvialmxfhsadsocmpuhwdkhefmtfbmbkianmpp
ivmcioowtxhsbyfcsvvsyfmzupwzjbpkbjfpjwdmeequbrwawmtgnqvrkpqst
cmatbmbjpwuipxeuwofuhekuhtmvuejipelgvifrcmatdggqvhsdgbnqawhgw
avjoffaapujadpb

你还知道 Alice 和 Bob 使用的是 Vigenere 加密法。请还原成明文。

7. 假设你截获了 Alice 和 Bob 的以下密文：

cotqv iwosr vcjop sleky xkpkr xrzao vclkw yskhf vkkav
gwnaw cestd zvmar wrzlq dsfdh kmmwe chyic yrewn rmazi
lipzy srrkq qlvax acuwg rbkwa dmoig oolwb pglwd peoem
ilvyi gwgnf dvwty uymsy glsws cyzxl tadhk aufyb kiicy
zctw

你只知道 Alice 和 Bob 使用的是某种自动密钥加密法。请还原成明文。

8. 请用圆柱面加密法将以下明文加密：

> Alice, how was your vacation? I hear that LA is really hot this time of year. Did you get all the rest your need? We have all missed you at work and are looking forward to your return on Monday. Yours, Bob

其中，圆柱面的顺序为：

20　1　19　2　18　3　17　4　16　5　15　6　14　7　13　8　12　9　11　10

密文的 IC 值为多少？对密文进行频率分析，并就结果进行说明。

9. 假设你截获了 Alice 和 Bob 的以下密文：

yncnz xrfls nnsln tgaac wruyv bsdjj nqrqj aeexf dfjoy
rbyeu eutcb vspst iifsx aujfj kdlun pvcaf ajazj ncadn
yutly aabnr pohaj jlxzx nixth bsbht fkppo bdlgu cnual
chunu ojbdt ltpsm euqpl twele

你还知道 Alice 和 Bob 使用的是圆柱面加密法，但不知道圆柱面的顺序。你猜测 Tuesday 一词出现在了其中。你也可以猜测其他单词。请还原成明文。

10. 使用有 3 个回转轮（依次为轮 1、2、3，起始位置为“A B C”）的回转轮加密法，将以下明文加密。

Alice, waiting until Tuesday is fine I hope the doctor's appointment does not mean that something is wrong. I think we need to explore other options for our ciphers. It is possible that Eve has made a major breakthrough. I no longer trust our current system. You know this is really becoming a pain. Yours, Bob

密文的 IC 值为多少？对密文进行频率分析，并就结果进行说明。

11. 使用如图 3.17 所示的 Enigma 加密机，轮 1 前转一步，重新绘制该图，并确定“a”将映射到何处。

12. Eve 截获了 Alice 和 Bob 的以下密文：

gcckk dejjk ubecv tsews byqrj xgydi vsjat pjpyc
oqveq keaoq tajqn ngzrs xybnq gxpgi ws

Eve 怀疑 Alice 和 Bob 使用的是回转轮加密法，且“alice”一词出现在了文字开头的某处。请还原明文，并指出密钥的排列方式。

13. 同音替换加密法与多码加密法的区别是什么？

14. 请尝试构建你自己的同音替换加密法，用 0～100 之间的数字来替代字母。

第 4 章 经典多图加密法

4.0 概 述

看到本章的标题，你可能会被“多”(poly)字弄糊涂了。如果是这样，没关系。为了简单起见，这里有必要中断一下，先来介绍我们在加密法世界中所处的位置。你也许还记得，在单码加密法中，每个字符是由另一个字符所替代；而在多码加密法中，每个明文字符可以用多个密文字符来替代。但是，单码和多码加密都是作用于单个字符。凡是一次加密一个字母的加密法都称为单图加密法(monographic cipher)。多图加密法(polygraphic cipher)则是作用于字符组。明文的 n 个字符组合被密文的 n 个字符组替代。最简单的例子是双图(digraphic)替换加密法，该加密法一次加密两个字符。例如，字符对“at”在密文中可能被“ui”替换，而字符对“an”则可能被“wq”替代，等等。双图加密法比单图加密法更安全，因为双图(即字母对)比单个字母要多。标准英语字母表只有 26 个字母，但有 676 种双图(26×26)。因此，通过其特征来确定双图更困难。这几乎可以使简单频率分析法失效。如果一次加密三个字母(即三图(trigraphic)加密法)，那么该加密法的功能就更大了，因此在英语中有 17 576 种三图(26×26×26)。

多图加密法就是决定如何将明文字母组映射到密文字母组。对于双图加密法，可以创建一个表格，列举出 676 种明文对及其相应的密文对。事实上，的确有用这种方法构建的加密法，但管理起来困难。而要为三图加密法构建这种表就太令人望而却步了。(如果真构建了这种表，那与其说这是一个加密法，倒不如说它就是一个密码。)我们需要的是使用某种算法，它基于密钥来生成多图对替换，就像在前面两章中介绍的加密法一样。不用奇怪，人们已经开发了一些这种加密法。这些加密法演示了现代的块加密法(后面章节将介绍)的基本特性。

4.1 Playfair 加密法

Bob 和 Alice 认为，在这场与 Eve 无结尾的战斗中，单码和多码加密法都不安全。他们回去查找资料并发现一种功能强大的双图加密法，称为 Playfair 加密法。Playfair 加密法被认为是双图替换加密法的基本范例，就像 Vigenere 加密法是多码加密法的基本范例一样。

Playfair 实际上是由 Charles Wheatstone 开发，他是 19 世纪时期的一个多产的发明家和科学家。这个加密法是以 Wheatstone 的朋友 Lyon Playfair 命名的。其中的部分原因是因为，该加密法是由 Baron Playfair 在 1854 年 1 月的宴会上介绍给英国政府的，并经过

努力使该加密法被英国政府接收。尽管 Playfair 从没有为自己声称过什么，但该加密还是以他的名字来命名的。其原因不太清楚，但有证据表明，英国在布尔战争期间使用了该加密法。

Playfair 加密法不仅比我们已经学习过的标准单码和多码加密法更具挑战性，而且它的实现令人出奇的简单。该加密法是基于一个关键词的，该关键词填写在一个 5×5 的正方形中(去除重复字母和字母 j)。字母表中其他未用到的字母按顺序填写在正方形的剩下空位中。例如，关键词“telegram”可构成如下的正方形：

t	e	l	g	r
a	m	b	c	d
f	h	i	k	n
o	p	q	s	u
v	w	x	y	z

通过在正方形中查找明文对，并根据以下三个规则之一选取密文，从而将明文字母对转换成密文对：

- 如果两个明文字符出现在正方形的同一行，选用右边的字符作为密文。
- 如果两个明文字符出现在正方形的同一列中，选用下方的字母作为密文。
- 如果两个明文字符出现在不同的行和列，那么它们就形成了包含这两点的正方形，选用在这个正方形上另外两个点的字母作为密文。

在应用这些规则之前，需要将明文进行一些处理，即去除字母“j”和所有双字母。用字母“i”代替所有的“j”，在双字母直接插入一个等效字母(通常是“x”或“q”)。另外，如果明文的字母数为奇数，则在末尾添加一个等效字母，使字母数为偶数。例如，使用前面介绍的关键词为“telegram”的 Playfair 方格，明文“next time, jay, try something different”的处理方法如下：

(1) 将字母“j”改为“i”，明文变成了“next time, iay, try something different”。

(2) 在双“t”和“different”的双“f”字母之间各插入一个“q”，明文变成了“next qtime, iay, try something difqferent”。

(3) 由于明文有 34 个字母，不需要在末尾添加字母“q”了。当然，现在的明文看起来有点奇怪的样子，但还是可读的。

(4) 对于明文的第一对字母“ne”，在如下表中查找，用这两个字母所在的点画一个四方形。该四方形另两个角上的字母就是这个明文对的相应密文，即“hr”。

t	e	l	g	r
a	m	b	c	d
f	h	i	k	n
o	p	q	s	u
v	w	x	y	z

(5) 重复该过程，可得出明文“xt”的密文为“vl”，“qt”的密文为“ol”。

最后生成的密文是“hrvlo lhblh cvetg ypale ikkrb nioht tlfr”。

尽管这个加密法用手工实现起来很简单，但很枯燥。因此，Bob 和 Alice 利用 CAP 软件来实现 Playfair 加密法的自动加密和解密。从 Cipher 菜单中选取 Playfair 项，将弹出如

图 4.1 所示的 Playfair Cipher 窗口。在该窗口中输入关键词，选择 Set Key 后，将出现 Playfair 方格。然后，只需要选取 Encipher 或 Decipher 就可以进行加密和解密了，其他的由软件自动完成。

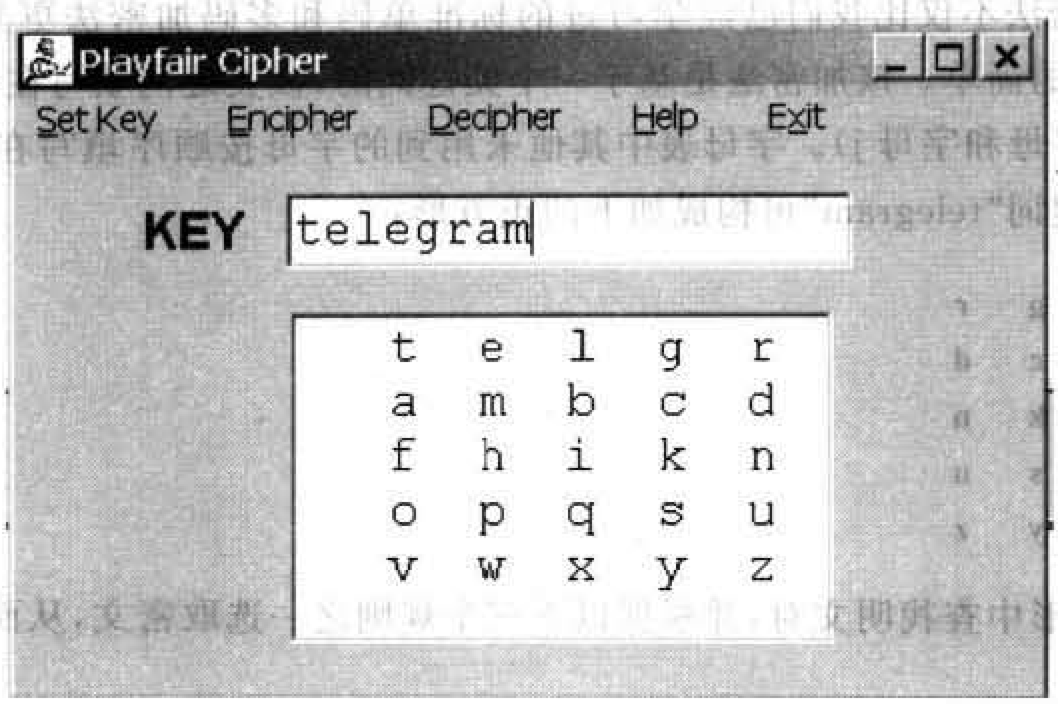


图 4.1　Playfair Cipher 窗口

4.1.1　Playfair 加密法分析

与前面介绍的加密法相比，Playfair 加密法功能很强大。但它不是没有弱点。事实证明有两种破解它的方法，一种是探测词破解法，另一种是关键词发现破解法。这两种方法都可以用来破解 Playfair 加密法。

破解的第一步是确认它的确是 Playfair 加密法。这可以通过查看 Playfair 的密文特征来做到。可以查看以下 4 个特征：

(1) 密文的字母数必须是偶数的。

(2) 明文中平时少见的辅音字母“j”、“k”、“q”、“x”和“z”，在密文中出现比较频繁。

(3) 如果分解成双图形式，不会出现如“ss”或“ee”等的重复字母。

(4) 双图的频率分布与明文的大致相同。

由于 Bob 和 Alice 选用 Playfair 作为他们的新加密法，Eve 开始时被所截获的消息弄糊涂了。最后，她发现 Bob 和 Alice 使用了一种完全不同的加密法。知道了 Bob 和 Alice 阅读了哪些书籍后，她怀疑这种新的加密法是 Playfair 加密法，因此她从截获的以下密文样本(已分成两个了)中查找前面介绍的那些特征：

pk km km ew dw qn bs hl uf gq zk zp tl fc ls fq tn ca zw ae
ns fq tn zw ps el kz kc xc rb ke tm wg co ab fk vn cl uf ui
df ch hq kc mp

正如 CAP 软件可以帮助 Bob 和 Alice 加密一样，它也可以对 Playfair 加密法执行测试。Eve 选取 Polygraphic Tools 菜单下的 PolyID 按钮，选中三个核选框就可以运行这三个测试。结果如图 4.2 所示。由于这个密文具有 Playfair 加密法的特征，Eve 决定尝试使用已知明文攻击法来破解，因为她认为短语“a sample of”出现在了明文之中。

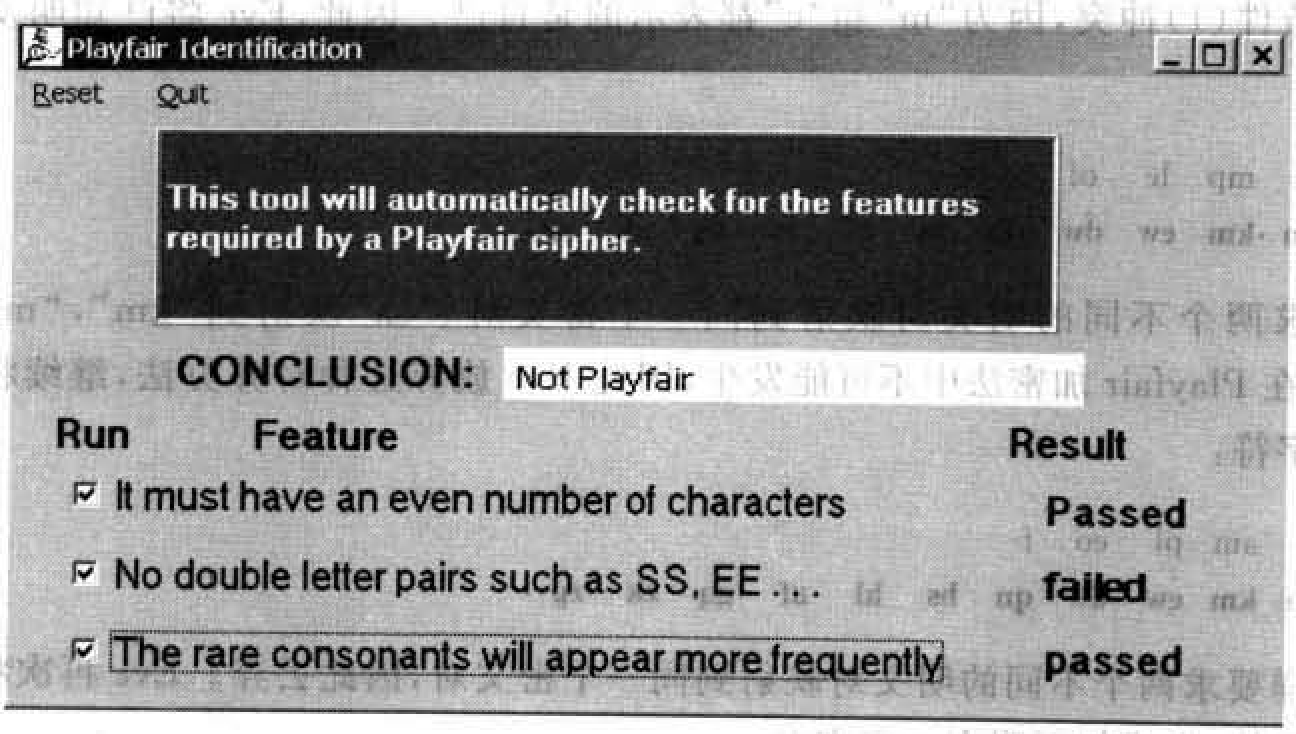


图 4.2 Playfair Identification 窗口

用已知明文攻击法破解 Playfair 加密法的一般步骤如图 4.3 所示。攻击法的第一步是找出与已知明文匹配的密文模式。这可以通过应用 Playfair 加密法的规则来实现，尤其是验证不会与 Playfair 加密法的任何一个特征冲突。这些特征包括如下：

(1) 明文中的每个字母在密文中都不会用其自己来表示，也就是说，明文字母"a"不可能用密文字母"a"来表示。

(2) 明文中两个倒置的双图(如"ER"和"RE")在密文中也是用倒置的双图来表示的。

(3) 明文的每个字母只能用某 5 个字母中的一个来加密，这 5 个可能的字母包括，在 Playfair 方格中位于该字母下方的字母以及其四周的其他 4 个字母。

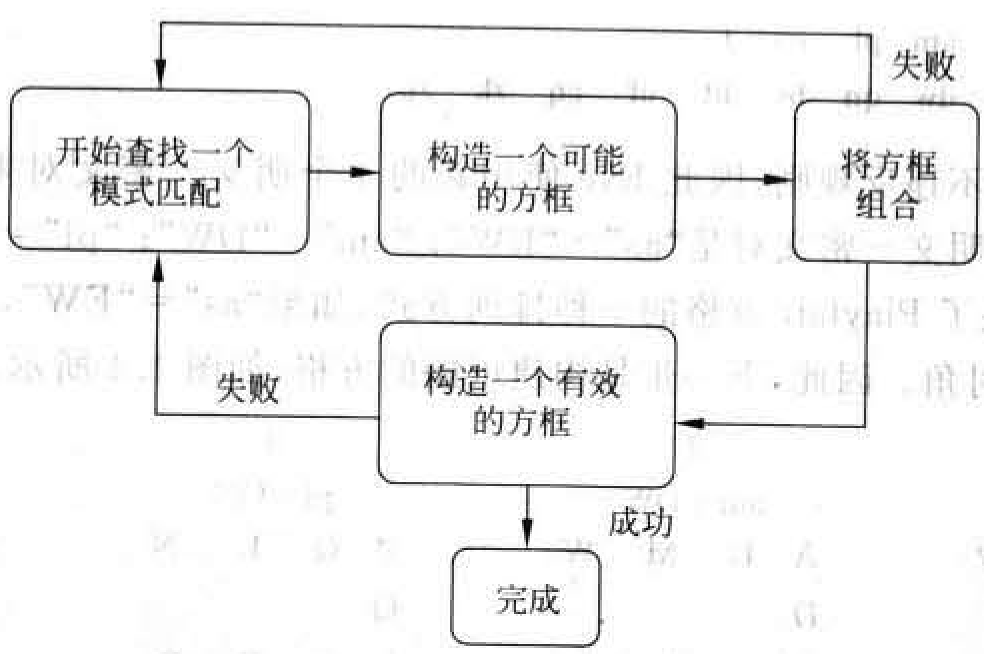


图 4.3 攻击 Playfair 加密法的步骤

如果当前已知明文－密文对与这些条件冲突，那么将已知密文移动一个字符，并重新检查这些条件。如果没有冲突，那么相应的密文就可能(仅仅是可能)表示已知明文。

Eve 首先将已知明文与密文的第一个字符对齐，如下所示：

```
as am pl eo f-
pk km km ew dw qn bs hl uf gq zk zp
```

这与条件(1)冲突，因为“m”和“e”都表示的是自己。因此，Eve 将已知明文后移一个字符：

-a sa mp le of
pk km km ew dw qn bs hl uf gq zk zp

这要求两个不同的明文对映射到同一个密文对(“sa”映射到“km”，“mp”映射到“km”)，这在 Playfair 加密法中不可能发生，因此 Eve 放弃这种对齐方法，继续将已知明文后移一个字符：

as am pl eo f-
pk km km ew dw qn bs hl uf gq zk zp

这同样要求两个不同的明文对映射到同一个密文对，因此丢弃。Eve 再次将已知明文右移一个字符，生成如下明文一密文对：

a sa mp le of
pk km km ew dw qn bs hl uf gq zk zp

这可能行了。不仅因为它符合规则，而且还可以看出，已知短语前的一个明文字母是“s”(你知道为什么吗?)。这可以生成 4 个可能的明文一密文对：“sa”=“KM”；“mp”=“EW”；“le”=“DW”；以及“of”=“QN”。这些明文一密文对可以用在下一步中，用来生成 Playfair 方格。但是，经过 Eve 的一些努力后，仍没有找到解决办法。Eve 于是返回到这一步，继续将已知明文右移一个字符，并重复上述尝试，都没有成功，直到已知明文与密文如下对齐：

as am pl eo f-
pk km km ew dw qn bs hl uf gq zk zp

这种对齐方式不违反规则，因此 Eve 使用新的 4 个明文一密文对来尝试生成 Playfair 方格。这 4 个新的明文一密文对是“as”=“EW”；“am”=“DW”；“pl”=“QN”；以及“eo”=“BS”。每对都反映了 Playfair 方格的一种排列方式，如果“as”=“EW”，那么“aesw”必须在同一行、同一列或对角。因此，下一步是构建可能的方格，如图 4.4 所示。

1

as=EW

A	E	S	W
E			
S		A	E
W		W	S

2

am=DW

A	D	M	W
D			
M		A	D
W		W	M

3

pl=QN

P	Q	L	N
Q			
L		P	Q
N		N	L

4

eo=BS

E	B	O	S
B			
O		E	B
S		S	O

图 4.4 由 4 个明文一密文对构建可能的 Playfair 方格

如果这些明文一密文对都正确，那么就应该可以将这 4 种排列方式融合到一个完整的方格中。在前面的尝试中，Eve 没能构建成一个统一的方格，但她看到了希望。例如，从方格 1 中可以看出，“a”和“e”必须在同一行或同一列上。方格 2 也要求“a”和“d”在同一行或同一列上。方格 4 要求“e”和“b”在同一行或同一列上。将这几种情况一起考虑，就可以知道“a”、“b”、“d”和“e”必须在同一行或同一列上。有了这些信息，就可能对 Playfair 方格的

排列结构做初步的设想。我们知道，在关键词填写完后，字母表的其他字母是按顺序填写到 Playfair 方格中的，因此“a”、“b”、“c”、“d”和“e”不是关键词的字母，且可能是一起出现在第二行上。Eve 发现这很有可能，于是顺着它继续下去（题外话，Eve 发现的这些是密码分析中一个最为重要的特征——跟着感觉走。）。至此，Playfair 方格的一部分内容如下所示：

-	-	-	-	-
A	B	C	D	E
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-

下面可以利用图 4.4 所示的各种可能性来完成关键词的构造。方格 1 和 2 都要求“w”和“a”在同一行或同一列。方格 4 要求“s”和“e”在同一行或同一列。方格 3 要求“d”和“m”在同一行或同一列。这些表明，Playfair 方格可能是如下所示：

W	-	-	M	S
A	B	C	D	E
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-

观察上面的结构后，Eve 猜测关键词可能是“Worms”，于是完整的 Playfair 方格就如下所示：

W	O	R	M	S
A	B	C	D	E
F	G	H	I	K
L	N	P	Q	T
U	V	X	Y	Z

Eve 尝试利用该关键词来解密密文获得成功，证明她找到了正确的关键词了。这也同时说明 Alice 和 Bob 的加密法又一次失败了。

即使 Eve 没有已知的明文段，也可以使用第一次世界大战时期开发的一种破解方法来找到该关键词。这种方法是由美国军方的无线智能服务（G. 2. A. 6）部门主管大弗兰克·穆尔曼提出的。该方法是通过猜测关键词的字母来破解 Playfair 加密法的。当然，一味地盲目猜测不是一个有效的方法，但大弗兰克·穆尔曼有他自己的想法。他利用关键词的一些特征来帮助猜测。他知道，在关键词中，每三个辅音字母就很可能有两个元音字母，而且关键词往往包含一些更为普通的字母。大弗兰克·穆尔曼还发现，如果密文中的某个字母大量地与其他字母组合，那么该字母就很可能是关键词的字母（因为关键词的字母在明文中使用更频繁）。利用以上这些，他就可以开发一种破解 Playfair 加密法的密文攻击法。

他的方法是分析密文，找出那些大量地与其他字母组合的字母集。这些字母很可能包含了大多数的关键词字母。将其他字母填入 Playfair 方格中，反复尝试，就可能找出关键词。

4.1.2 用 CAP 软件分析 Playfair 加密法

与往常一样，CAP 软件为你提供了一些用来破解 Playfair 加密法的基本工具。一旦知道了使用的是 Playfair 加密法（这可以利用 CAP 软件的 Playfair ID 工具来确定），就很有可能破解它。如果知道了一个可能的单词，CAP 软件的 Playfair Known Word 工具就可在密文中查找可能的匹配。例如，在图 4.5 中，CAP 软件在密文的第 5 个字母处为单词“plaintext”找到了一个可能的匹配。利用这 4 对字母，就可以设置 Playfair 方格的合适排列，然后再用它来验证其他的匹配，并用如图 4.6 所示的 Playfair Worksheet 来查找其他可能的单词。如果第一个匹配不行，CAP 软件列出了所有可能的匹配，因此可以使用第二个匹配。诚然，这需要花费一些时间和具有一定的耐心，但 Playfair 加密法的密钥是对这种努力的回报。

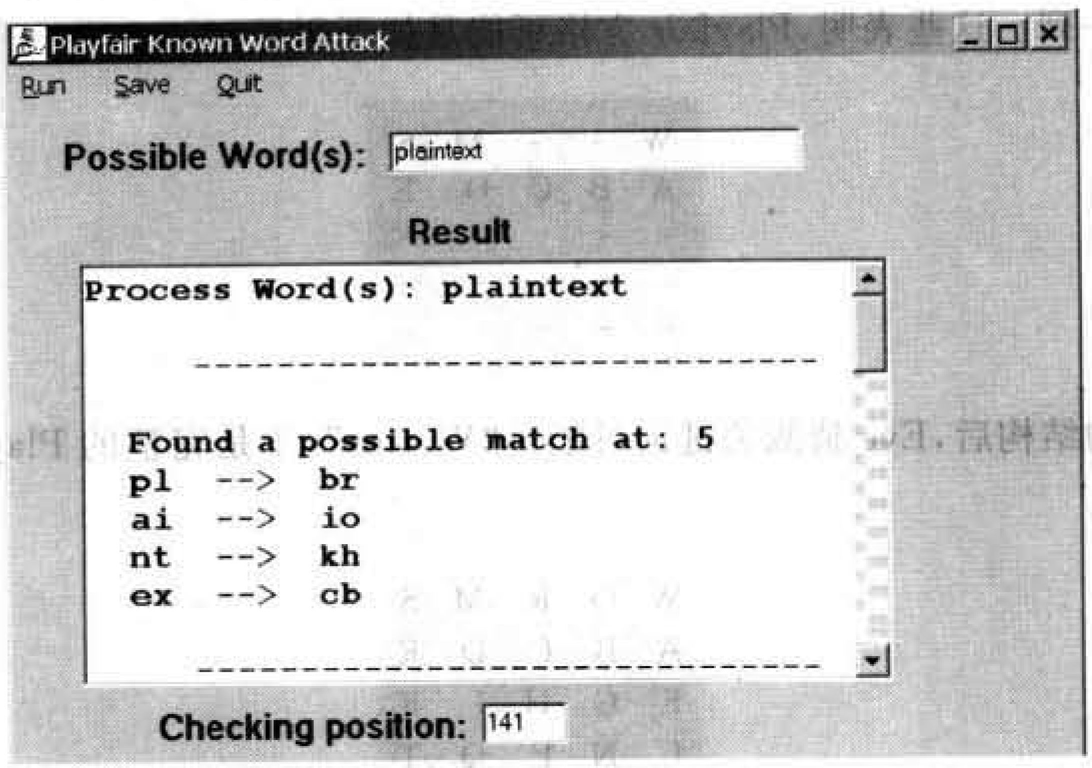


图 4.5 Playfair Known Word Attack 窗口

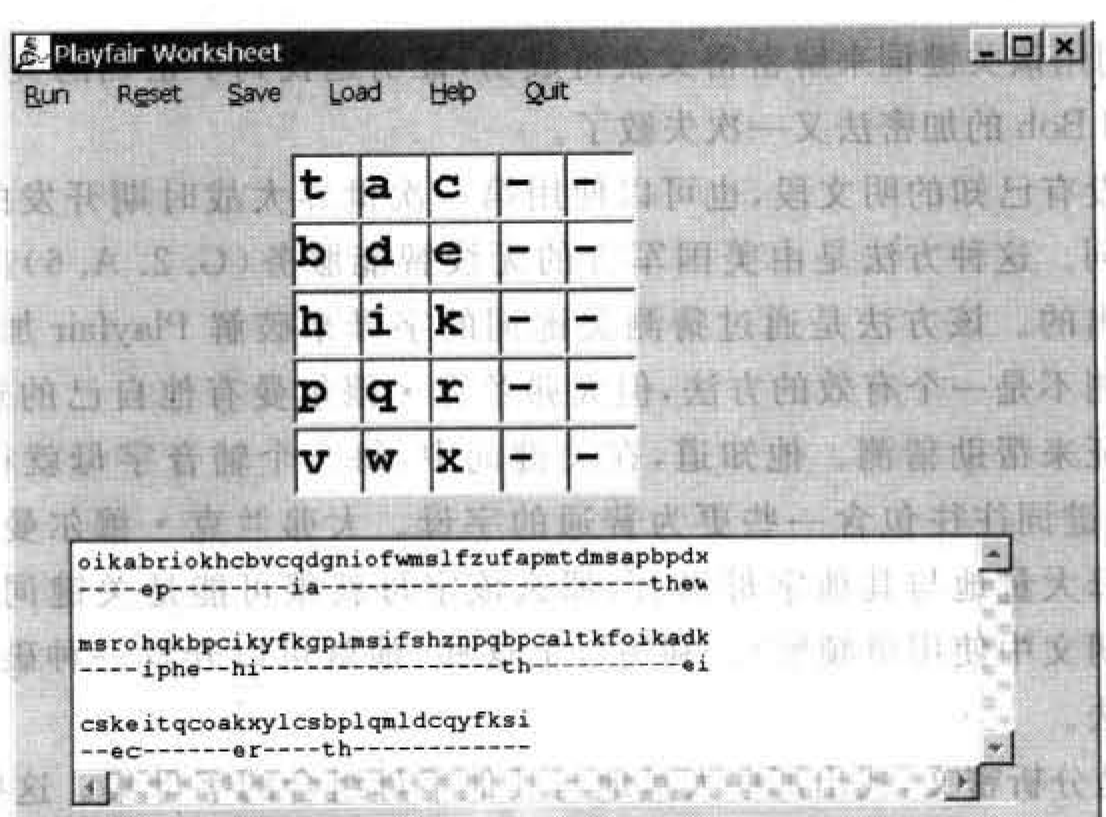


图 4.6 Playfair Worksheet 窗口

4.2 Hill 加密法

Playfair 加密法强调的是一次加密多个字母。一旦 Playfair 加密法为大家所熟悉后，密码学家开始尝试设计三连字或更多字的加密法。他们的努力失败了，部分原因是要维护三维表是很困难的。要成功使用多图的加密法，需要将数学方法应用到密码学中去。

用数学方法来创建常见的多码加密法，首次出现在 1929 年的 Hill 加密法中。该加密法是以其发明人 Lester S. Hill 来命名的。他是纽约 Hunter 学院的数学教授。1929 年，他发表了一篇名为"Cryptography in an Algebraic Alphabet"的短论文，提出了一种基于联立方程的加密算法。该加密算法将含有 m 个字母的明文块加密成含有 m 个字母的密文块。每个明文字母被赋予一个数值，通常是 $a=0$，$b=1$，…，$z=25$，但 Hill 使用的是随机赋值。块中每个字母的数值一起用来生成一组新的数值，这些数值就用来表示密文字母。例如，如果 $m=3$，那么 3 个明文字母的数值（假设为 p_1、p_2 和 p_3）将通过如下的方程组转换成密文数值 c_1、c_2 和 c_3：

$$c_1 = (k_{11}p_1 + k_{12}p_2 + k_{13}p_3) \bmod 26$$
$$c_2 = (k_{21}p_1 + k_{22}p_2 + k_{23}p_3) \bmod 26$$
$$c_3 = (k_{31}p_1 + k_{32}p_2 + k_{33}p_3) \bmod 26$$

该加密法的密钥是 k_{ij} 值。取模操作的意思是将和除以 26，得余数。

一种方法是由如下的方程组给定的：

$$c_1 = (17p_1 + 17p_2 + 5p_3) \bmod 26$$
$$c_2 = (21p_1 + 18p_2 + 21p_3) \bmod 26$$
$$c_3 = (2p_1 + 2p_2 + 19p_3) \bmod 26$$

利用这三个方程，明文"now"首先转换成数字形式：13 14 22，将这些数值代入方程组得出密文的数值：23 20 4。将这些数值转换回字母后，就是密文"xue"了。

最简单的方法是使用矩阵。加密密钥是矩阵 M，而解密密钥是 M^{-1}。对于前面的示例，密钥为：

$$M = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \qquad M^{-1} = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix}$$

$$\begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \begin{bmatrix} 13 \\ 14 \\ 22 \end{bmatrix} \bmod 26 = \begin{bmatrix} 23 \\ 20 \\ 4 \end{bmatrix}$$

要使这个过程可行，则密钥矩阵必须是可逆的，因此密钥值不可以随机选取。可以从 $m=3$的示例中归纳出用数学方法表示 Hill 加密法的一般形式。密钥写成一个 $m \times m$ 的可逆矩阵形式：

$$K = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1m} \\ k_{21} & k_{22} & \cdots & k_{2m} \\ & & \cdots & \\ k_{m1} & k_{m2} & \cdots & k_{mm} \end{bmatrix}$$

将明文分成块，每个块含义 m 个字母，用 $m\times 1$ 的向量表示。例如，第 i 块含有字符 p_1，p_2，…，p_m，写成如下形式：

$$b_i = \begin{bmatrix} p_1 \\ p_2 \\ \cdots \\ p_m \end{bmatrix}$$

那么密文就由如下计算结果确定：

$$Kb_i = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1m} \\ k_{21} & k_{22} & \cdots & k_{2m} \\ & & \cdots & \\ k_{m1} & k_{m2} & \cdots & k_{mm} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \cdots \\ p_m \end{bmatrix} \bmod 26 = \begin{bmatrix} c_1 \\ c_2 \\ \cdots \\ c_m \end{bmatrix}$$

4.2.1 在 CAP 软件中实现 Hill 加密法

Alice 和 Bob 都不愿意花时间去开发自己的 Hill 加密法密钥(他们不想去测试每个可能的密钥是否可逆)，因此他们借助 CAP 软件来生成可逆的随机密钥。CAP 可以随时满足他们的需要。要利用 CAP 的这个特征，只需在 CAP 的 Cipher 菜单中选取 Hill Cipher 项，将弹出如图 4.7 所示的窗口。首先设置矩阵的大小，然后选取 Key 菜单的 Autokey 项，生成一个可逆的新矩阵密钥。该密钥可留作后面使用。一旦有了密钥后，就可以像其他加密法一样，在 CAP 软件中生成密文或明文，只需单击 Encipher 或 Decipher 即可。当你在 CAP 软件中运行 Hill 加密法时，将会注意到有的数字大于 25。这是因为 CAP 软件使用的是 ASCII 码的十进制值来表示每个字符的。由于 ASCII 是 8 位的，因此在 CAP 软件中，Hill 矩阵的数字大小为 0～255。

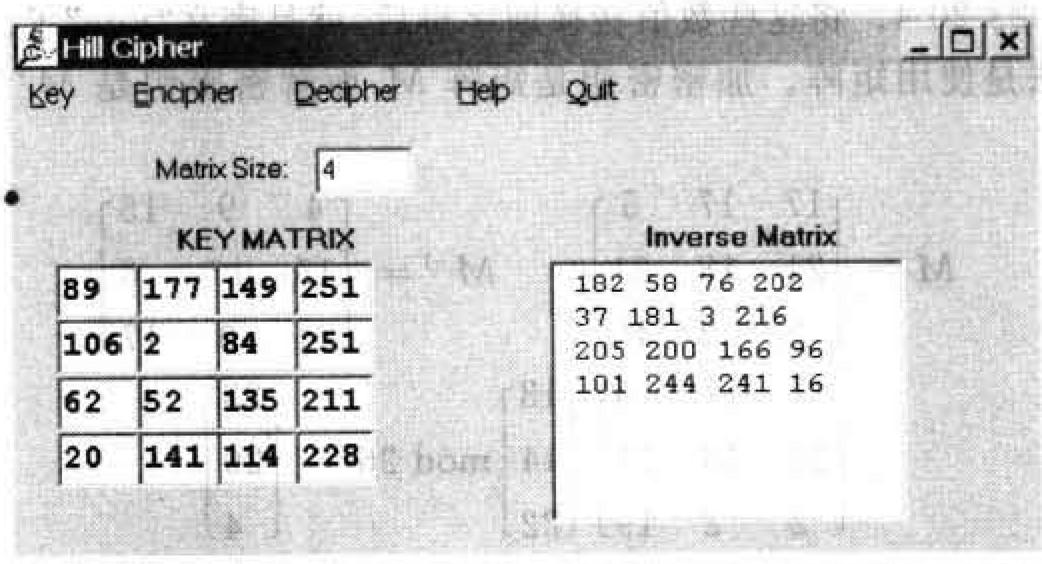


图 4.7 Hill Cipher 窗口

4.2.2 Hill 加密法分析

Hill 加密法可以很好地防止密文攻击。事实上，其矩阵越大，该加密法抗攻击能力就越强。但是，使用已知明文攻击法可以很容易地破解该加密法。其攻击过程很像用来破解仿

射加密法的方法，即用已知明文-密文组建立方程组，求解该方程组后，就可找出其密钥。

例如，假设 Alice 和 Bob 使用的是密钥矩阵大小为 4 的 Hill 加密法，加密了一些明文。Eve 截获了以下一些密文（只是一个数字流），并怀疑是经 Hill 加密法加密的：

209	37	59	255	110	196	247	90	52	34	1	68	11	130	43
23	20	26	219	119	93	164	12	63	110	202	124	137	112	158
232	23	127	118	128	123	115	89	62	224	199	10	199	142	104
242	120	4	142	26	230	159	129	164	133	153	31	256	210	62

知道了 Alice 和 Bob 是借助 CAP 软件加密的后，Eve 只需要两条信息就可以破解该密文：矩阵大小和一些已知明文。她非常怀疑密文含有这样一些单词："she can not attack us"。她不知道矩阵的大小，但她知道应在 2～6 之间，因为这是 CAP 软件的限制，因此她尝试每个矩阵大小。当她尝试矩阵大小为 4 时，她将这个已知明文短语转换到 4×4 的矩阵 M 中。她知道，如果该短语是用这个密钥矩阵 K 加密的，那么结果密文矩阵 C 就应是 M 和 K 的乘积。也就是：

$$C = M \times K$$

因此，由于知道了 C 和 M，Eve 就可以如下解出 K：

$$K = M^{-1} \times C$$

当然，Eve 并不知道该短语加密成了密文的哪一段。因此，她不得不为每部分密文生成一个可能的密钥，并应用于整个密文，直到发现某个密钥可以真正生成明文。在上面的示例中，将已知明文短语写成如下的矩阵：

$$M = \begin{bmatrix} 115 & 104 & 101 & 99 \\ 97 & 110 & 110 & 111 \\ 116 & 97 & 116 & 116 \\ 97 & 99 & 107 & 117 \end{bmatrix}$$

该矩阵的逆阵为（选取 Polygraphic Analysis 工具下的 Matrix Ops，CAP 软件可以为你生成该逆阵）：

$$M^{-1} = \begin{bmatrix} 254 & 104 & 110 & 99 \\ 236 & 236 & 195 & 75 \\ 224 & 10 & 109 & 18 \\ 202 & 52 & 155 & 61 \end{bmatrix}$$

接下来就是将该逆阵与密文矩阵相乘。每个结果都可能是密钥，可以将这些可能的密钥来解密密文。大多数时候，这些密钥是行不通的，但其中有一个是可以的。Eve 发现的可行密钥是：

$$K = \begin{bmatrix} 4 & 6 & 154 & 69 \\ 129 & 34 & 179 & 103 \\ 26 & 32 & 78 & 121 \\ 41 & 22 & 30 & 54 \end{bmatrix}$$

这个过程如果由手工来完成可能会很困难，但和往常一样，CAP 软件提供了必要的工具。选取 Polygraphic Tools 菜单下的 Hill Attack 按钮，将弹出如图 4.8 所示的窗口。输

入你认为最可能的矩阵大小,在明文中可能出现的单词。当你单击 Run 时,CAP 软件将列出(通常是非常长的)各种可能的密钥矩阵,以及用这些密钥解密的文字。从这些解密的文字中找一个可读的,如果有,那么你就找出了密钥。

这个过程有两个限制。第一是必须有足够的已知明文来生成一个矩阵。第二是被测试的明文段必须匹配矩阵的边界值。

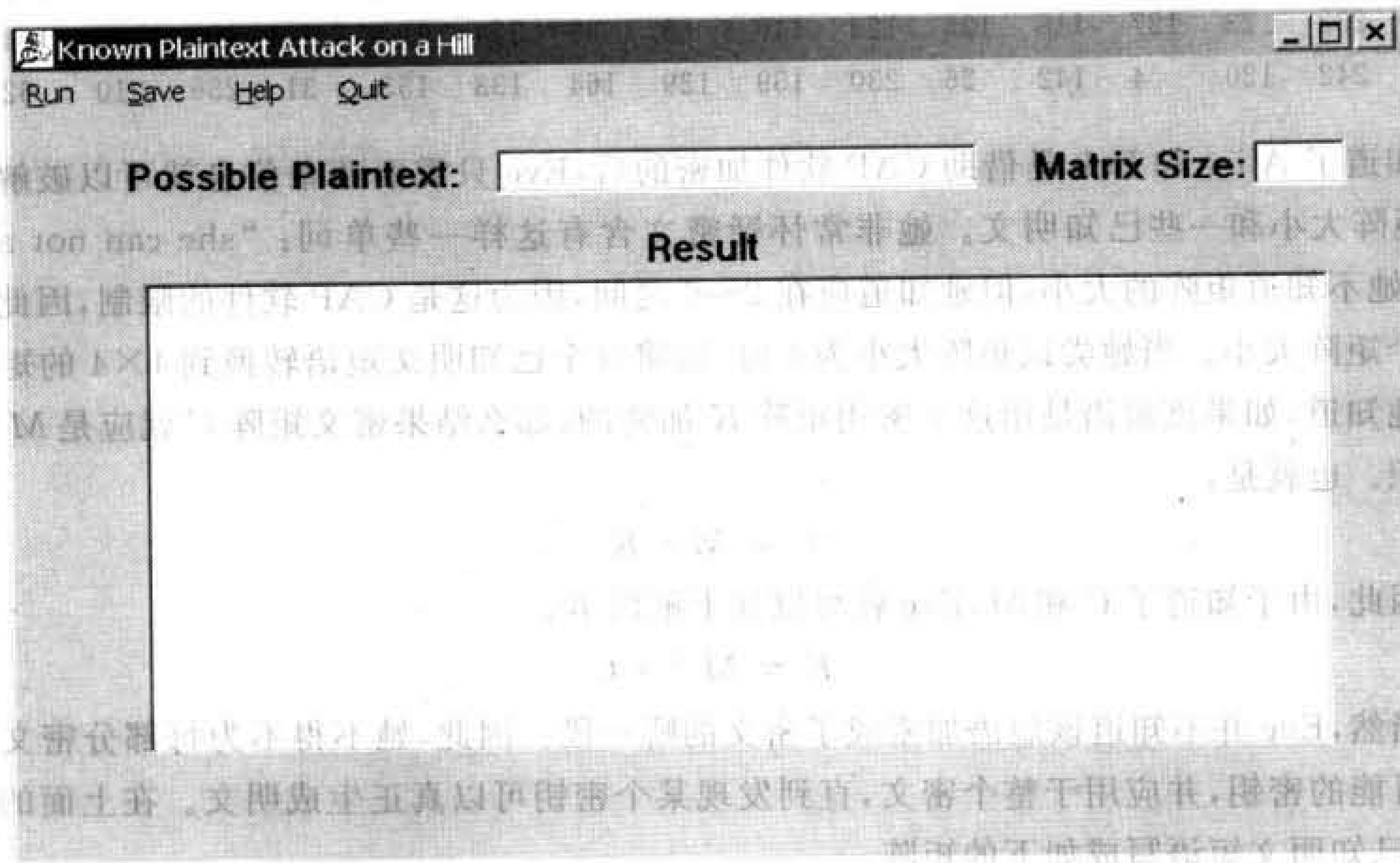


图 4.8 CAP 的 Known Plaintext Attack on a Hill 窗口

4.3 Beale 加密法的历史简介

尽管 Beale 加密法的密文有些是多图的,但严格地说,Beale 加密法不属于多图加密法。这里之所以介绍 Beale 加密法,是为了说明个人对密码学与密码分析的影响。通过 Beale 加密法的介绍,使你对密码学的内部工作原理有一个更基本的理解。

Robert Morris 是维吉尼亚客栈的管理员,他砸开了一个保存了近 20 年的盒子。这个盒子是 Morris 的一个朋友 Thomas Jefferson Beale 在 1821 年给他的,Beale 告诉 Morris,如果他 10 年内没有回来,就打开它。自 Beale 离开以后,Morris 就再也没有他的任何消息。当 Morris 打开盒子时,他被震惊了。盒子里有两封写给他的信,以及一些写满数字的纸。信中说,1817 年,Beale 和他的 30 个朋友在圣达菲以北 250 英里处发现了一个特大的金银矿。在两年的时间中,Beale 和他的一些朋友回到了维吉尼亚,埋藏了一吨的金银。在写满数字的纸上,隐藏着埋藏金子的地点,以及与他一起工作的 30 个人的后代的姓名。请 Morris 去挖出这些黄金,并将黄金分给名单上的人。信中还说,加密密钥将邮寄给他,但他从没有收到过。

Morris 带着他无法读懂的宝藏图离开了。多年来他一直企图破解这些密码,但没有取得任何进展。最后,他告诉了 James Ward,并请他来分析这些密文。有意思的是,Ward 从

这三段密文的第二段中发现了密钥。第二段密文是由 1～1322 的数字组成的：

115,73,24,807,37,52,49,17,31,62,647,22,7,15,140,47,29,
107,79,84,56,239,10,26,811,5,196,308,85,52,160,136,59,
211,36,9,46,316,554,122,106,95,53,58,2,42,7,35,122,53,
31,82,77,250,196,56,96,118,71,140,287,28,353,37,1005,65,
147,807,24,3,8,12,47,43,59,807,45,316,101,41,78,154,
1005,122,138,191,16,77,49,102,57,72,34,73,85,35,371,59,
196,81,92,191,106,273,60,394,620,270,220,106,388,287,63,
3,6,191,122,43,234,400,106,290,314,47,48,81,96,26,115,
92,158,191,110,77,85,197,46,10,113,140,353,48,120,106,2,
607,61,420,811,29,125,14,20,37,105,28,248,16,159,7,35,
19,301,125,110,486,287,98,117,511,62,51,220,37,113,140,
807,138,540,8,44,287,388,117,18,79,344,34,20,59,511,548,
107,603,220,7,66,154,41,20,50,6,575,122,154,248,110,61,
52,33,30,5,38,8,14,84,57,540,217,115,71,29,84,63,43,
131,29,138,47,73,239,540,52,53,79,118,51,44,63,196,12,
239,112,3,49,79,353,105,56,371,557,211,505,125,360,133,
143,101,15,284,540,252,14,205,140,344,26,811,138,115,48,
73,34,205,316,607,63,220,7,52,150,44,52,16,40,37,158,
807,37,121,12,95,10,15,35,12,131,62,115,102,807,49,53,
135,138,30,31,62,67,41,85,63,10,106,807,138,8,113,20,32,
33,37,353,287,140,47,85,50,37,49,47,64,6,7,71,33,4,43,
47,63,1,27,600,208,230,15,191,246,85,94,511,2,270,20,39,
7,33,44,22,40,7,10,3,811,106,44,486,230,353,211,200,31,
10,38,140,297,61,603,320,302,666,287,2,44,33,32,511,548,
10,6,250,557,246,53,37,52,83,47,320,38,33,807,7,44,30,
31,250,10,15,35,106,160,113,31,102,406,230,540,320,29,66,
33,101,807,138,301,316,353,320,220,37,52,28,540,320,33,8,
48,107,50,811,7,2,113,73,16,125,11,110,67,102,807,33,59,
81,158,38,43,581,138,19,85,400,38,43,77,14,27,8,47,138,
63,140,44,35,22,177,106,250,314,217,2,10,7,1005,4,20,25,
44,48,7,26,46,110,230,807,191,34,112,147,44,110,121,125,
96,41,51,50,140,56,47,152,540,63,807,28,42,250,138,582,
98,643,32,107,140,112,26,85,138,540,53,20,125,371,38,36,
10,52,118,136,102,420,150,112,71,14,20,7,24,18,12,807,
37,67,110,62,33,21,95,220,511,102,811,30,83,84,305,620,
15,2,10,8,220,106,353,105,106,60,275,72,8,50,205,185,
112,125,540,65,106,807,138,96,110,16,73,33,807,150,409,
400,50,154,285,96,106,316,270,205,101,811,400,8,44,37,52,
40,241,34,205,38,16,46,47,85,24,44,15,64,73,138,807,85,
78,110,33,420,505,53,37,38,22,31,10,110,106,101,140,15,
38,3,5,44,7,98,287,135,150,96,33,84,125,807,191,96,511,
118,40,370,643,466,106,41,107,603,220,275,30,150,105,49,
53,287,250,208,134,7,53,12,47,85,63,138,110,21,112,140,
485,486,505,14,73,84,575,1005,150,200,16,42,5,4,25,42,8,
16,811,125,160,32,205,603,807,81,96,405,41,600,136,14,20,
28,26,353,302,246,8,131,160,140,84,440,42,16,811,40,67,
101,102,194,138,205,51,63,241,540,122,8,10,63,140,47,48,
140,288

Ward 发现，这些数字表示的是美国《独立宣言》中的一些单词。每个明文字母对应于这些加编号的单词的第一个字母。结果明文如下：

I have deposited in the country of Bedford about four miles from Bufords in an excavation or vault six feet below the surface of the ground the following articles belonging jointly to the parties whose names are given in number three herewith, the first deposit consisted of ten hundred and fourteen pounds of gold and thirty eight hundred and twelve pounds of silver deposited nov eighteen nineteen. The second was made dec eighteen twenty one and consisted of nineteen hundred and seven pounds of gold and twelve hundred and eighty eight of silver also jewels obtained in St Louis in exchange to save transposition and valued at thirteen thousand dollars. The vault is roughly lined with stone and the vessels rest on solid stone and are covered with others. Paper number one describes the exact locality of the value so that no difficulty will be had in finding it.

关键(即包含宝藏埋藏地点的)密文在第一张纸中，具体如下：

71,194,38,1701,89,76,11,83,1629,48,94,63,132,16,111,95,
84,341,975,14,40,64,27,81,139,213,63,90,1120,8,153,126,
2018,40,74,758,485,604,230,436,664,582,150,251,284,308,
231,124,211,486,225,401,370,11,101,305,139,189,17,33,88,
208,193,145,1,94,73,416,918,263,28,500,538,356,117,136,
219,27,176,130,10,460,25,485,18,436,65,84,200,283,118,
320,138,36,416,280,15,71,224,961,44,16,401,39,88,61,304,
12,21,24,283,134,92,63,246,486,682,7,219,184,360,780,18,
64,463,474,131,160,79,73,440,95,18,64,581,34,69,128,367,
460,17,81,12,103,820,62,116,97,103,862,70,60,1317,471,
540,208,121,890,346,36,150,59,568,614,13,120,63,219,812,
2160,1780,99,35,18,21,136,872,15,28,170,884,30,44,112,
18,147,436,195,320,37,122,113,6,140,8,120,305,42,58,461,
44,106,301,13,408,680,93,86,116,530,82,568,9,102,38,416,
89,71,216,728,965,818,2,38,121,195,14,326,148,234,18,55,
131,234,361,824,5,81,623,48,961,19,26,33,10,1101,365,92,
88,181,275,346,201,206,86,36,219,324,829,840,64,326,19,
48,122,85,216,284,919,861,326,985,233,64,68,232,431,960,
50,29,81,216,321,603,14,612,81,360,36,51,62,194,78,60,
200,314,676,112,4,28,18,61,136,247,819,921,1060,464,895,
10,6,66,119,38,41,49,602,423,962,302,294,875,78,14,23,
111,109,62,31,501,823,216,280,34,24,150,1000,162,286,19,
21,17,340,19,242,31,86,234,140,607,115,33,191,67,104,86,
52,88,16,80,121,67,95,122,216,548,96,11,201,77,364,218,
65,667,890,236,154,211,10,98,34,119,56,216,119,71,218,
1164,1496,1817,51,39,210,36,3,19,540,232,22,141,617,84,
290,80,46,207,411,150,29,38,46,172,85,194,39,261,543,897,
624,18,212,416,127,931,19,4,63,96,12,101,418,16,140,230,
460,538,19,27,88,612,1431,90,716,275,74,83,11,426,89,72,
84,1300,1706,814,221,132,40,102,34,868,975,1101,84,16,79,
23,16,81,122,324,403,912,227,936,447,55,86,34,43,212,107,
96,314,264,1065,323,428,601,203,124,95,216,814,2906,654,
820,2,301,112,176,213,71,87,96,202,35,10,2,41,17,84,221,
736,820,214,11,60,760

Ward 发现美国《独立宣言》对这段密文无效。他又尝试了宪法、圣经、莎士比亚作品以及其他的文字，没有一个能有效。一些人放弃了努力，认为这是一个骗局。他们也许是对的。其他一些人声称破解了，但没有一个能找到宝藏。

不管真假与否，Beale 密码消耗了很多人的生命时间。1966 年，一位银行家投资买了一台锄耕机，打了大量的孔，但一无所获，甚至推平了一座山丘也没有找到任何东西。看来埋藏黄金的神秘地点和秘密的加密法对我们的想象还是很有吸引力的，直至耗尽我们的生命。

4.4 本章小结

本章介绍了多图加密法的概念，也就是说，一次加密两个或多个字符的加密法。结果是这种加密法可以抵抗简单频率分析攻击。但是，这些加密法并不是就无懈可击了。已知明文攻击法就可以破解这种加密法。

Alice 和 Bob 与 Eve 之间的又一轮斗争的结果是，谁也没有取得决定性的优势。Alice 和 Bob 可以使用 Playfair 加密法和 Hill 加密法，在一段时间内阻止 Eve 阅读他们的信息。但是，Eve 最终发现了这些多图加密法的弱点，并借助已有的技术，破解了 Alice 和 Bob 之间的通信。

他们都从这个保护信息的过程中获得了新的知识。Alice 和 Bob 发现了一种新的加密法，他们从该加密法中看到了一些前景。他们也再一次懂得了复杂并不意味着安全。现在，他们知道了使用替换方法要保持通信安全很困难，因此他们准备尝试一些完全不同的加密法（将在第 5 章中介绍）。

Eve 发现了已知明文攻击法的威力了。当密码分析的标准工具走人死胡同时，猜测出一部分的明文非常有用。她还发现，使用加密法的微小弱点，并不能反映明文字符本身。当 Alice 和 Bob 使用 Playfair 加密法时，对 Alice 和 Bob 来说是一个小事情，但对 Eve 该加密法却有帮助。像以往一样，从多图加密法破解的经历中，Eve 更加懂得了耐心和毅力的重要性。

总之，本章介绍了两种新的加密法，加入到加密法结构图中就如图 4.9 所示了。

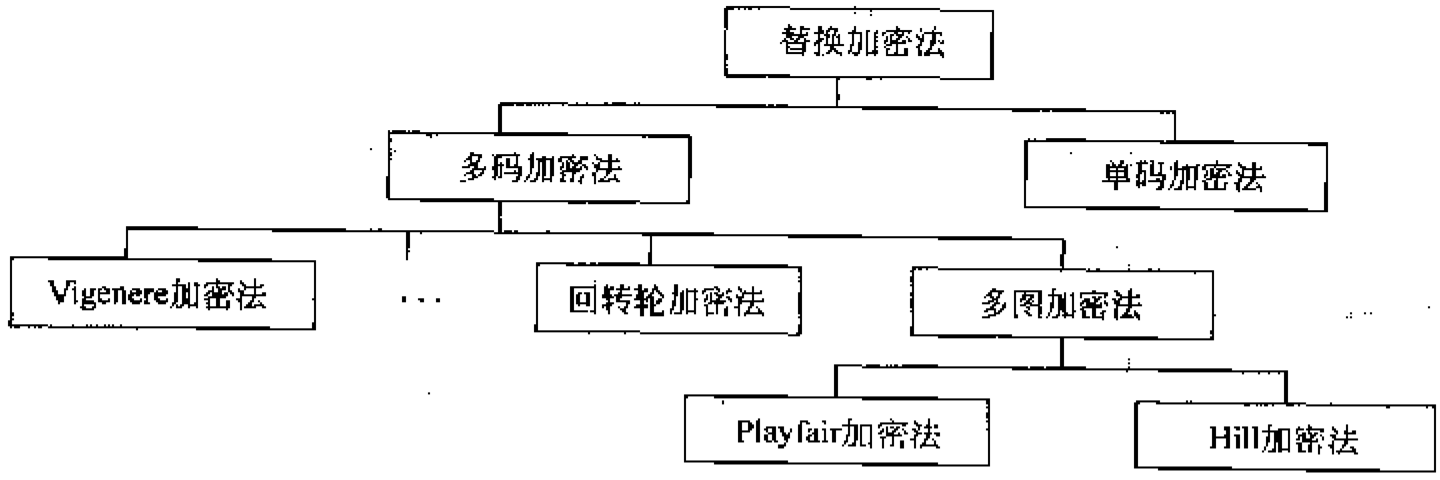


图 4.9　多图加密法

4.5 重要术语

Hill 加密法　　一种基于矩阵相乘的多图加密法。
Playfair 加密法　　一种使用一个关键词方格来加密字符对的加密法。
多图加密法　　一种一次将两个或多个字符加密的加密法。

习　题

1. 利用本章中所介绍的 Hill 加密法矩阵示例，查找"new"的密文，然后将它与"now"的密文进行比较。说明比较的结果。

2. 填写以下信息。

加密法	操作过程概述	弱点概述
Hill 加密法		
Playfair 加密法		

3. 创建关键词为"classroom"的 Playfair 加密法，并用它来加密以下文字：

Bob, I find this Playfair cipher easy to use but when it is deciphered I don't like all the j's and the q's. Sure I can read the plaintext just fine but it looks so strange that I wonder if some mistake has been made. How do you feel about the deciphered text? Does it bother you as much as it bothers me? Yours, Alice

找出这段明文的 IC，并在其上运行单个字母出现频率分析。解释你所查看到的。

4. 利用如下的 3×3 Hill 加密法密钥，将问题 3 中的明文加密。密钥的逆是多少？找出密文的 IC，并同样在其上运行单个字母出现频率分析。解释你所查看到的。

113	13	197
97	131	126
146	219	50

5. 找出以下 Playfair 密文的明文（已知单词"frequency"在该文字中）：

gkfam	omyug	cqrup	hdsau	zhoic	tgnqs	pltrq	ihcuf	iqmwp
kuklh	haelq	tauaz	hpiqr	tgpxm	azlux	tcfop	gdaco	fbkia
ytdqa	lnaos	typhm	ecvbi	lsecu	plugm	ctqpo	pzdmy	uflhp
cxfyg	poeas	mlhli	ceau					

6. 找出以下 Hill 密文的明文（已知短语"no more secrets"在该文字中）：

129	228	145	221	241	236	190	156	27
107	71	193	29	30	226	250	191	54

141	210	111	31	106	1	105	30	50
234	0	180	159	121	149	46	218	11
21	142	133	118	124	220	191	3	18
65	203	133	219	17	14	188	100	181
94	197	149	225	40	213	179	92	190
190	99	147	229	9	176	160	207	87

7. Playfair 加密法的密钥数有多少？如果编写一个程序，用计算机来察看这些密钥，每个密钥耗时 10^{-6} 秒，那么使用强力攻击法破解 Playfair 加密法平均耗时多少？

8. 问题 7 多估计了密钥的数量，因为大多数 Playfair 加密法的用户更愿意选用一个单词作为密钥。假设 Playfair 加密法的密钥是从有 10 000 个单词的字典中选取的，那么密钥的数量是多少，用问题 7 中介绍的程序来破解该加密法需要多长的时间？

9. 如果关键词可以从任意位置开始填写在矩阵中，那么 Playfair 加密法的功能就更强了。基于这种假设重做问题 8。请提出至少一种能增加 Playfair 加密法的密钥数量的方法。

第 5 章　经典换位加密法

5.0 概　　述

Alice 和 Bob 疲惫不堪地使用各种替换加密法来保护其通信，但都失败了，于是他们认为需要尝试一些完全不同的加密法。在查阅了有关资料后，他们发现了一种新的加密法（至少在他们看来是新的），这种加密法根本不使用替换，这就是**换位加密法**（transposition cipher）。换位加密法不是用其他字母来替代已有字母，而是重新排列文本中的字母。这种加密法有点像拼图游戏。在拼图游戏中，所有的图块都在这里，但排列的位置不正确。换位加密法开发者的目标是，设计一种方法，使你在知道密钥的情况下，能将图块很容易地正确排序。而如果没有这个密钥，就不可能解决。当然，密码分析员的目标是在没有密钥的情况下重组拼图，或从拼图的特征中发现密钥。这两种目标都很难实现。

第 1 章已经介绍过，古斯巴达克人使用的 skytale 加密法就是早期的一种换位加密法。尽管换位加密法的历史不是太清楚，但这种类型的加密法经常被使用。在美国国内战争时期，北方军就使用过一种换位加密法，用来将电报消息的内容加密，以防止同盟军的窃听。

北方军使用的加密法是使用单词换位，而不是单个字母换位。先将明文写成一行一行，然后按列转录。例如，明文消息可能含有这样一些内容："We will attack at down with the first division on the right and the second division on the left. Surprise is important, so make all preparations under the cover of darkness"。这些消息可以写成 7 列：

We	will	attack	at	down	with	the
first	division	on	the	right	and	the
second	division	on	the	left	Surprise	is
important	so	make	all	preparations	under	the
cover	of	darkness	nulls	fill	up	the

如果最后一行不够，用空字符填满。因此，如果以这样的顺序读取列：第一列从下往上读，第三列从上往下读，第四列从下往上读，第二列从上往下读，第六列从下往上读，第七列从上往下读，第五列从下往上读，那么，密文为：

"cover important second first we attack on on make darkness nulls all the the at will division so of up under surprise and with the the is the the fill preparations left right down"

某些单词出现在密文中可能会泄漏其含义，为了提高加密法的性能，一些字母可能被其他字母替换。因此，在前面的示例中，"attack"可能会被"blue"替代。而且还开发了更复杂的模式。美国国内战争时期，南方军使用的是非常脆弱的加密法，如恺撒加密法和只有 3 个不同密钥的 Vigenere 加密法。

有两种常见的换位加密法，它们的区别在于分解单位的不同。单图（monographic）换位

使用的是单个字母，例如斯巴达克人使用的 skytale 加密法；而多图(polygraphic)换位使用的是多个字母(如单词或短语)，例如北方军使用的加密法。在这两种方法中，单图换位往往功能更强大，也是本章讨论的重点。

换位加密法使用的密钥通常是一些几何图形，它决定了重新排列字母的方式。例如，在第 1 章中已经介绍过的古希腊人使用的 skytale 加密法。这种加密法的密钥是 skytale 杆，就是用它来打乱字母的顺序的。其他使用的几何图形包括正方形和矩形。明文字母按一个方向填写(例如，skytale 加密法是从上往下)，而从另一个方向读取(skytale 加密法是将纸卷起后从左到右读取)。

rail-fence 加密法不是按从上到下的方式填写明文和读取密文的，而是使用了对角线方式。在这种加密法中，明文是按 Z 字形的方式填写在矩形的对角线上，而按行读取生成密文。例如，如果矩形的高为 3，长为 11，那么明文"this is a test"在该矩形中的填写如图 5.1 所示。而按行读取所生成的密文就是"tiehsstsiat"。

t				i				e		
	h		s		s		t		s	
		i				a				t

图 5.1　rail-fence 加密法示例

同样的过程可以应用于其他几何图形。例如，在一个固定大小的矩形中，可以将明文填写成一个三角形，然后按列读取生成密文。图 5.2 所示的是将明文"You must do that now"填写在一个 7×4 的矩形中。按行读取生成的密文是"tuhosayuttmdnoow"。

			y			
		o	u	m		
	u	s	t	d	o	
t	h	a	t	n	o	w

图 5.2　三角形方式的换位

5.1　置换加密法

换位加密法的一种简单实现是置换加密法。这很像洗一副纸牌。在置换加密法中，将密文分成了固定长度的块，如 d，置换函数 f 用于从 1～d 中选取一个整数。然后，每个块中的字母根据 f 重新排列。这种加密法的密钥就是(d, f)。例如假设 d 为 4，f 给定为(2, 4, 1, 3)。这意味着第 1 个字符移到位置 2，第 2 个字符移到位置 4，第 3 个字符移到位置 1，第 4 个字符移到位置 3。利用这种置换加密法将明文"codes and ciphers are fun"加密，首先将明文分成块，每块包含 4 个字母，即"code sand ciph ersa refu nxxx"。注意最后一个块要用空字符填满。现在，就可以根据给定的函数 f 将每个块重新排列。对于第 1 个块，字母"c"排到位置 2，字母"o"排到位置 4，字母"d"排到位置 1，字母"e"排到位置 3，生成的密文是"dceo"。对每个块重复这个过程，最后生成的密文为"dceonsdapchipchisearfruexnxx"。

Alice 和 Bob 对置换加密法的简单性及其显而易见的强大功能印象很深，因此决定尝试一下。最后，他们有了一种用单字符频率分析法无法破解的加密法（至少他们是这样认为的）。他们查看 CAP 软件，发现该软件提供了这种加密法的实现。

使用 CAP 软件很容易，但是有某些限制。在 CAP 软件中，块的最大大小为 10。当 Alice 和 Bob 需要使用置换加密法时，只需输入明文，然后从 Cipher 菜单中选取 Permutation 项即可。此时将弹出一个窗口，让他们输入密钥，如图 5.3 所示。首先输入块的大小，然后输入置换规则。图 5.4 为置换加密法的一个示例，其中，块大小为 7。置换规则是将第 7 个字符排到位置 1，将第 3 个字符排到位置 2，将第 6 个字符排到位置 3，等等。

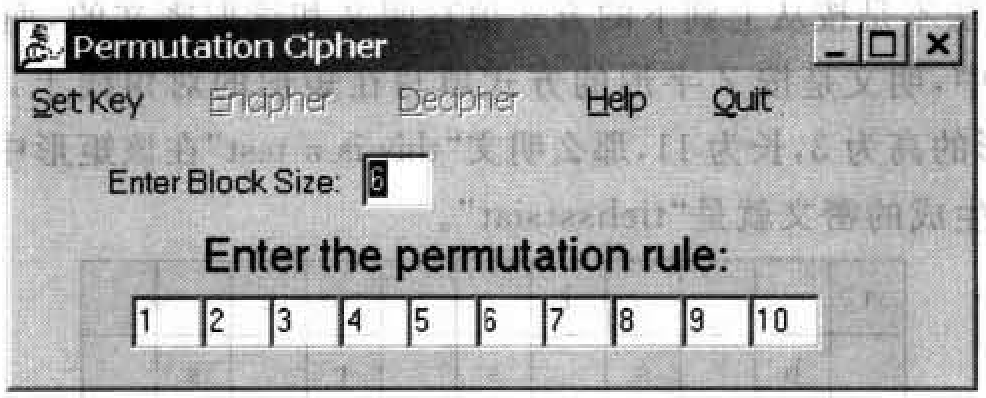


图 5.3 初始的 Permutation Cipher 窗口

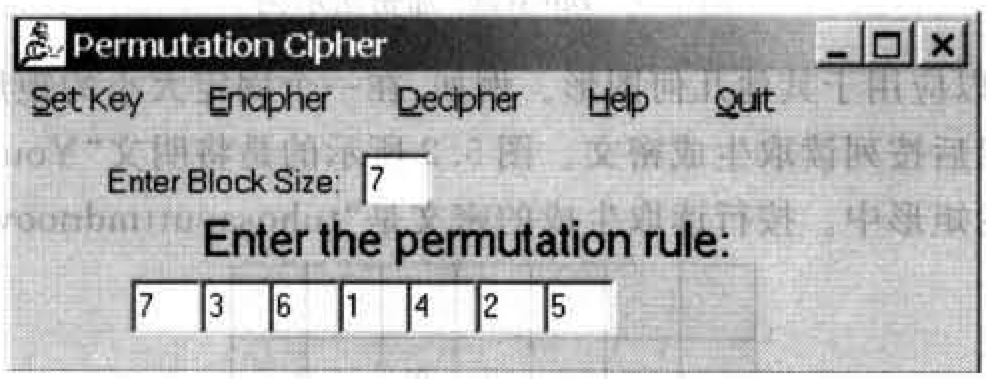


图 5.4 置换加密法密钥示例

5.1.1 置换加密法分析

可以用密文攻击法和已知明文攻击法来破解置换加密法。密文攻击法是查看密文块，查找出可能生成可读单词的排列方式。一旦发现了某个块的置换方式，就可以应用到密文的所有块中。如果密码分析员熟悉从颠倒字母顺序来构成单词，破解置换加密法就是一个简单的工作了。

已知明文攻击法就更简单。知道了明文中可能包含的一个单词，该攻击法有以下三个步骤：

（1）找出包含有与已知单词相同字母的块；

（2）通过比较已知单词与密文块，确定置换方式；

（3）在密文的其他块上测试上面得出的置换方式。

Eve 使用已知明文攻击法来攻击以下密文：

eialciithsansaehitcpoierhwipetzkors

Eve 怀疑单词“Alice”出现在了明文中，因此她用这个单词来匹配各块(块大小为 5)。如果某个密文块同样有这些字母，那它就可以重构该置换。在上面密文中，Eve 在密文的开头就找到了，因为第一个块包含有单词“alice”的所有字母。

```
alice
eialc
```

从这个匹配中，Eve 可以知道，第 1 个字符移到了位置 3，第 2 个字符移到了位置 4，第 3 个字符移到位置 2，第 4 个字符移到位置 5，第 5 个字符移到位置 1。将这个置换方式应用到密文的其他字母，就可以还原明文了。

在这个示例中，Alice 和 Bob 选择了一个糟糕的置换密钥。如果块的大小再大一些，也许该加密法就更强大些。块的大小增大，肯定能使手工查找匹配方式更困难。但是，即使是更大些的块，CAP 软件也可以帮助 Eve 来完成破解。

要使用 CAP 软件来破解置换加密法，在 Analysis Tools 中选取 Transposition，并单击 Permutation 按钮，弹出如图 5.5 所示的窗口。输入已知单词，CAP 将返回可能的替换。

CAP 软件还提供了密文攻击工作表，允许用户输入可能的块大小，并使用从颠倒字母顺序来构成单词的技术，来发现实际的置换方式。

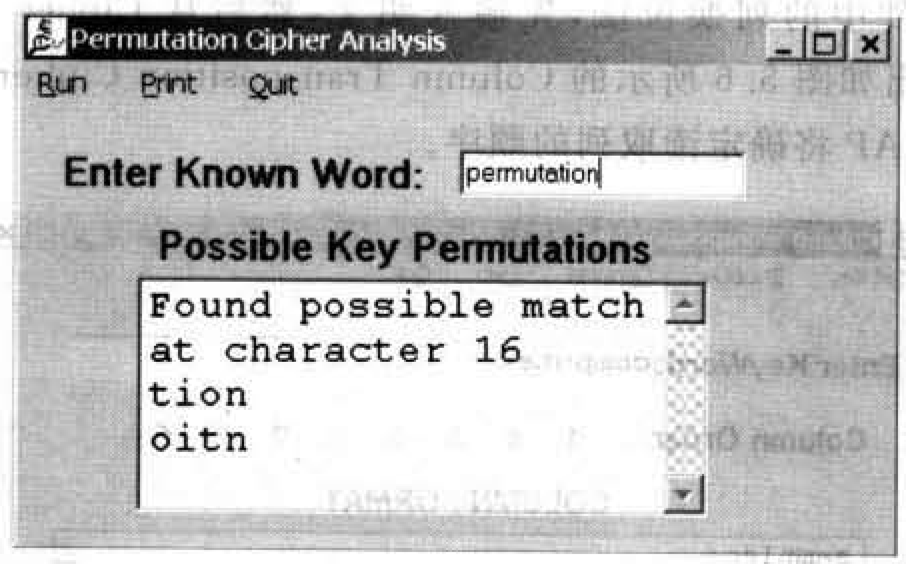


图 5.5　Permutation Cipher Analysis 窗口

5.2　列置换加密法

尽管 Eve 破解了置换加密法，Alice 和 Bob 还是看到了换位加密法的希望。因此他们决定使用一种更复杂的方案，一次包含更多的明文。他们选择了列置换加密法。利用这种加密法，明文按行填写在一个矩形中，而密文则是以预定的顺序按列读取生成的。例如，如果矩形是 4 列 5 行，那么短语“encryption algorithms”可以如下写入该矩形中：

1	2	3	4
e	n	c	r
y	p	t	i
o	n	a	l
g	o	r	i
t	h	m	s

按一定的顺序读取列以生成密文。对于这个示例，如果读取顺序是 4、1、2、3，那么密文就是："rilis eyogt npnoh ctarm"。这种加密法要求填满矩形，因此，如果明文的字母不够，可以添加"x"或"q"无效字母。

这种加密法的密钥是列数和读取列的顺序。如果列数很多，记起来可能会比较困难，因此可以将它表示成一个关键词，该关键词的长度等于列数，而其字母顺序决定读取列的顺序。

例如，关键词"general"有 7 个字母，意味着矩形有 7 列。由于"a"是"general"中字母顺序最低的，因此数字 1 放在第 6 列；第一个"e"为其次，所以数字 2 放在第 2 列；第二个"e"则是使数字 3 放在第 4 列。最后的结果如下：

```
g e n e r a l
4 2 6 3 7 1 5
```

因此由关键词"general"给定的读取列顺序为：4 2 6 3 7 1 5。

CAP 软件提供了一个完全填充的列换位算法。所谓"完全填充"就是矩阵的每个位置都必须有字符，即是最后一行必须填满。如果明文字母不够，就添加空字符来填满最后一行。

要使用 CAP 软件中的列换位法，先输入明文，然后从 Cipher 菜单中选取 Column Transposition 项，弹出如图 5.6 所示的 Column Transposition Cipher 窗口。注意，你所需要输入的是关键词，CAP 将确定读取列的顺序。

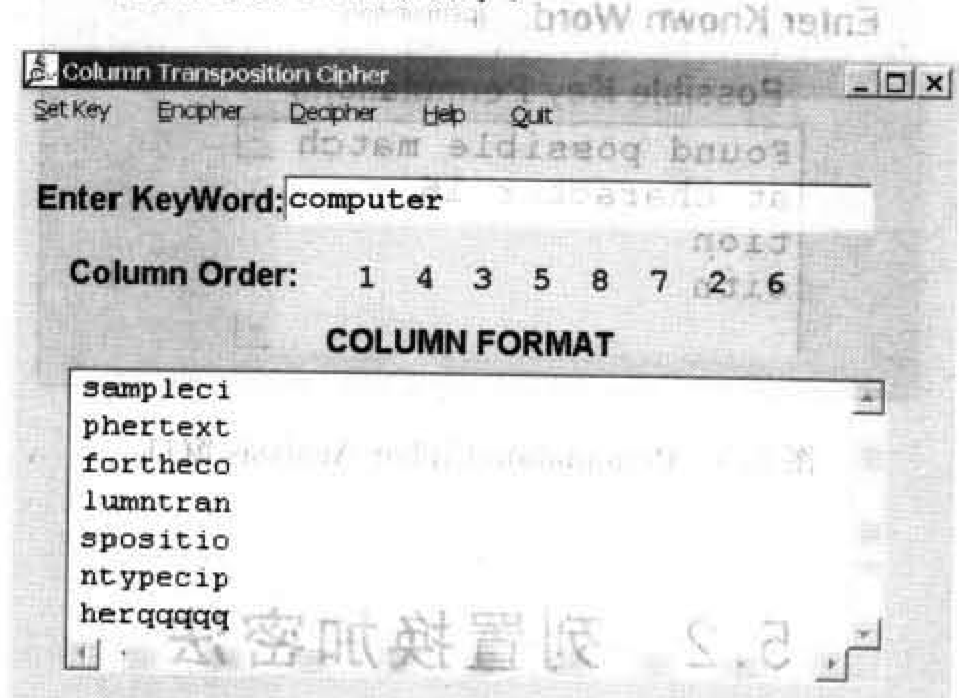


图 5.6 Column Transposition Cipher 窗口

5.2.1 列换位加密法分析

和往常一样，当 Alice 和 Bob 使用新加密法时都给 Eve 带来困难。她必须找到破解列换位加密法的方法，但她发现直接破解该加密法不可行。强力攻击法就是尝试所有可能的列和行，这付出很多，但收获甚微。她需要将这个问题分解。她决定以三步来处理列换位加密法。首先，尝试找出换位矩形的可能大小（多少行和多少列？）。第二，尝试找出这些可能的矩形中哪个是正确的。第三，知道了正确的矩形后，尝试重新排列矩形列，以便还原消息。

通过用这三个步骤来攻击，她又可以阅读 Alice 和 Bob 相互发送的消息了。

5.2.1.1 列的可能大小

这是攻击法的第一步，也是最简单的工作。因为该换位加密法完全是列换行，因此密文字符的数目必须是行数乘以列数的积(如果明文跨了多个矩形，那么就是该数的因数)。例如，假设 Eve 截获了以下消息：

NETEF LTDSR TSSTF MDCET DRHXS WHOHO EEADU OUUFI RRRRS
NEROT CFIEM EDSHA RTCPJ AOEGE WNLHO EPMWA WERUV AAINA
TSDDS OEOAC EHNTL HFLAU RAEEN OTOTS SOSYS TNNCG EMETT
YDYRR NEOOE RESTH INR

这个消息有 153 个字符。153 可以分解为 3×51、51×3、17×9 或 9×17。假设这个消息是在一个矩形中换位的，这 4 个积肯定定义了其大小，也就是说，这个矩形有 3 行 51 列、51 行 3 列、17 行 9 列或 9 行 17 列。没有其他矩形可以完全填满这 153 个字符了。由于 3×51 或 51×3 的列和行相差较大，不太可能是加密用的矩形。最有可能的是 17×9 或 9×17。因此，下一步就是找出这两者之间哪个是正确的。如果这两者都不正确，那么 Eve 就可以返回去尝试其他两个可能的大小。

5.2.1.2 确定正确的矩形

Eve 已经将范围缩小到了以上两种可能的矩形(如果都不行，那么她就得返回去尝试 3×51 或 51×3 了，如果是这样，那可是很不好的选择)。Eve 现在面临的问题是这两个中哪一个是正确的。她可以尝试重新排列这两个矩形的列，看看哪个可以生成有意义的单词，但她的工作多半是浪费在错误的矩形上。有效的方法是先找出最可能的矩形，然后再将努力放在这个矩形上。事实证明，要评估每个矩形是可能的，从而可以轻松地发现最可能的一个。

这个过程是基于这样一个事实，即矩形的每行表示的是标准英语的一行。明文的所有字母仍出现在密文中，它们只是错位了而已。因此 Eve 可以依靠英语的常见属性来检测密文的最可能排列。明文英语的每行应包含大约 40%的元音字母。如果某个矩形的元音字母分布满足每行 40%的标准，那么它就很可能是。

对于 9×17 的矩形，有 9 个列，因此每行应有大约 3.6(9×0.4)个元音字母。Eve 将密文填入这个矩形中，并计算每行的元音字母数目。她计算出实际的元音字母数与期望的元音字母数(3.6)之差的绝对值，并将这些差值相加，生成该矩形的得分。其计算结果如下：

1	2	3	4	5	6	7	8	9	元音字母数	差值
N	C	U	F	G	A	N	S	E	2	1.6
E	E	O	I	E	A	T	S	Y	7	3.4
T	T	U	E	W	I	L	O	R	4	0.4
E	D	U	M	N	N	H	S	R	2	1.6
F	R	F	E	L	A	F	Y	N	3	0.6
L	H	I	D	H	T	L	S	E	2	1.6
T	X	R	S	O	S	A	T	O	3	0.6

1	2	3	4	5	6	7	8	9	元音字母数	差值
D	S	R	H	E	D	U	N	O	3	0.6
S	W	R	A	P	D	R	N	E	2	1.6
R	H	R	R	M	S	A	C	R	1	2.6
T	O	S	T	W	O	E	G	E	4	0.4
S	H	N	C	A	E	E	E	S	4	0.4
S	O	E	P	W	O	N	M	T	3	0.6
T	E	R	J	E	A	O	E	H	5	1.4
F	E	O	A	R	C	T	T	I	4	0.4
M	A	T	O	U	E	O	T	N	5	1.4
D	D	C	E	V	H	T	Y	R	2	1.6
									总计	20.6

对于 17×9 的矩形，Eve 重复上面过程：

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	元音字母数	差值
N	R	E	O	U	N	M	P	L	E	T	C	A	O	N	Y	E	8	1.2
E	T	T	H	U	E	E	J	H	R	S	E	U	T	N	D	R	6	0.8
T	S	D	O	F	R	D	A	O	U	D	H	R	S	C	Y	E	6	0.8
E	S	R	E	I	O	S	O	E	V	D	N	A	S	G	R	S	7	0.2
F	T	H	E	R	T	H	E	P	A	S	T	E	O	E	R	T	6	0.8
L	F	X	A	R	C	A	G	M	A	O	L	E	S	M	N	H	5	1.8
T	M	S	D	R	F	R	E	W	I	E	H	N	Y	E	E	I	7	0.2
D	D	W	U	R	I	T	W	A	N	O	F	O	S	T	O	N	6	0.8
S	C	H	C	S	E	C	N	W	A	A	L	T	T	T	O	R	5	1.8
																	总计	8.4

最好的得分是差值总和最小，因此 17×9 的矩形看来是最好的选择。

5.2.1.3 还原列的顺序

破解列换位加密法的最后一步是找出列的正确顺序。这是从颠倒字母顺序而构成词的过程，需要充分利用字母的一些特征。一个很有用的帮助是"引导字母(pilot letter)"。有一些字母后面只能跟有限的某些字母。例如，在英语中，字母"q"后总是跟"u"。其他规则有：

- "j"后面只能跟元音字母；
- "x"前面只能是元音字母，除非它在词的末尾，它的后面只能接元音字母或"c"、"h"、"p"和"t"。

还有其他一些规则，可能也会有用。例如，某些字母(通常是中频或低频的)与其他字母组合后形成高频的：

- "h"(中频的)与"t"字母组合后形成"th"(高频的)；
- "h"与"c"(中频的)组合后形成"ch"。

要从颠倒字母顺序而构成词，先将密文填写到矩形中，然后查找引导字母或其他可能的字母对。在上面示例中，密文为：

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	元音字母	差值
N	R	E	O	U	N	M	P	L	E	T	C	A	O	N	Y	E	8	1.2
E	T	T	H	U	E	E	J	H	R	S	E	U	T	N	D	R	6	0.8
T	S	T	O	F	R	D	A	O	U	D	H	R	S	C	Y	E	6	0.8
E	S	R	E	I	O	S	O	E	V	D	N	A	S	G	R	S	7	0.2
F	T	H	E	R	T	H	E	P	A	S	T	E	O	E	R	T	6	0.8
L	F	X	A	R	C	A	G	M	A	O	L	E	S	M	N	H	5	1.8
T	M	S	D	R	F	R	E	W	I	E	H	N	Y	E	E	I	7	0.2
D	D	W	U	R	I	T	W	A	N	O	F	O	S	T	O	N	6	0.8
S	C	H	O	S	E	C	N	W	A	A	L	T	T	T	O	R	5	1.8
																	总计	8.4

在密文中没有“q”,但在第 2 行第 8 列有“j”。“j”必须与元音字母匹配,最可能的是第 5 列和第 13 列的“u”。由于这是一个方法尝试的过程,Eve 尝试将第 8 列与第 5 列匹配,然后与第 13 列匹配。其他行应生成可行的字母对,配好一对后继续重复这一工作。这里有一个问题是:“她如何从主观上确定哪个配对是最好的?”这个问题可以用百分比加权表来解决。

美国政府研究了 5000 连字集,并编辑了一个表格,该表格记录了每个连字频率的 2 倍数值。加权值越大,连字出现越多。Eve 选取加权值最大的两列,得出如下结果:

8	5	等级	8	13	等级
P	U	33	P	A	61
J	U	25	J	U	25
A	F	38	A	R	82
O	I	42	O	A	48
E	R	94	E	E	81
G	R	42	G	E	61
E	R	94	E	N	99
W	N	25	W	O	67
N	S	71	N	T	93
		464			617

从这个表格中,Eve 得出最可能的匹配是第 8 和 13 列。接下来的任务是找出位于这两列之前和之后的列。这可以通过继续查找匹配对来实现。例如,第 8 和 13 列所形成的“JU”对后面应跟一个辅音(最好是“n”或“s”),因此,第 15 和 11 列有可能:

8	13	15	等级	8	13	11	等级
P	A	N	89	P	A	T	83
J	U	N	68	J	U	S	58
A	R	C	53	A	R	D	64
O	A	G	45	O	A	D	73
E	E	E	81	E	E	S	86
G	E	M	61	G	E	O	58
E	N	E	87	E	N	E	87
W	O	T	67	W	O	O	45
N	T	T	67	N	T	A	74
			618				628

利用加权值，可以看出，第 13 与 11 列配对更可能些。第 8、13、15 列的第 5 行有 3 个“e”连着，这种情况不可能出现，因此更能确定是第 13 与 11 列配对。但是，如果万一第 13 与 11 列的配对失败，就应回去尝试第 8、13 和 15 列的组合。选定了 3 个列后，就容易开始查找可能的单词了，也可以同样计算加权值。在上面示例中，可以通过查看已确定的 3 个列，来看看其后应跟的列：

8	13	11	1	2	3	4	5	6	7	9	10	12	14	15	16	17
P	A	T	N	R	E	O	U	N	M	L	E	C	O	N	Y	E
J	U	S	E	T	T	H	U	E	E	H	R	E	T	N	D	R
A	R	D	T	S	D	O	F	R	D	O	U	H	S	C	Y	E
O	A	D	E	S	R	E	I	O	S	E	V	N	S	G	R	S
E	E	S	F	T	H	E	R	T	H	P	A	T	O	E	R	T
G	E	O	L	F	X	A	R	C	A	M	A	L	S	M	N	H
E	N	E	T	M	S	D	R	F	R	W	I	H	Y	E	E	I
W	O	O	D	D	W	U	R	I	T	A	N	F	S	T	O	N
N	T	A	S	C	H	O	S	E	C	W	A	L	T	T	O	R

第 2 列可能跟在第 11 列后，因为在第 2 行可以生成单词“just”，在第 7 行可以生成单词“enemy”。经过一定的努力后，Eve 最终破解了这个消息（这里留给读者去找出最终的消息）。

5.2.2 使用 CAP 软件来破解列换位加密法

用手工来破解列换位加密法可能比较困难，但是在 CAP 软件的帮助下，你所要做的只是单击鼠标了。在 CAP 主窗口的 Analysis 工具栏中，选取 Transposition 按钮，其中有好几个工具可用。第一个是 Vowel Analysis 选项，它可弹出如图 5.7 所示的窗口。它将自动计算密文的字符数，并报告可能的换位矩形大小。要执行元音字母分析，只要输入列的大小并单击 Run 即可。从 Vowel Results Summary 框中可以发现列大小的最佳值。

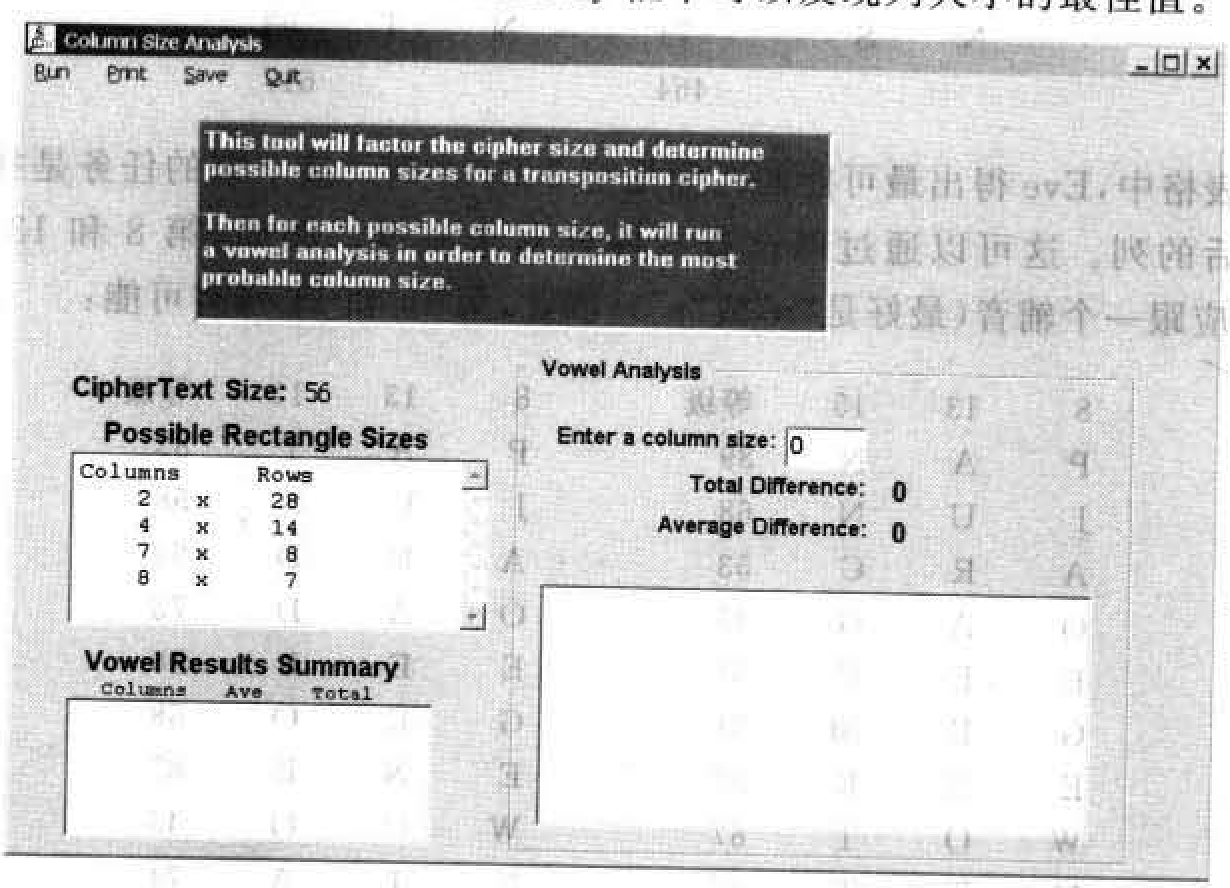


图 5.7 Column Size Analysis 窗口

从颠倒字母顺序构成词的过程以及加权值测试，都是 CAP 软件的工具集的一部分。从 Analysis 栏中选取 Anagram，CAP 将打开如图 5.8 所示的窗口。

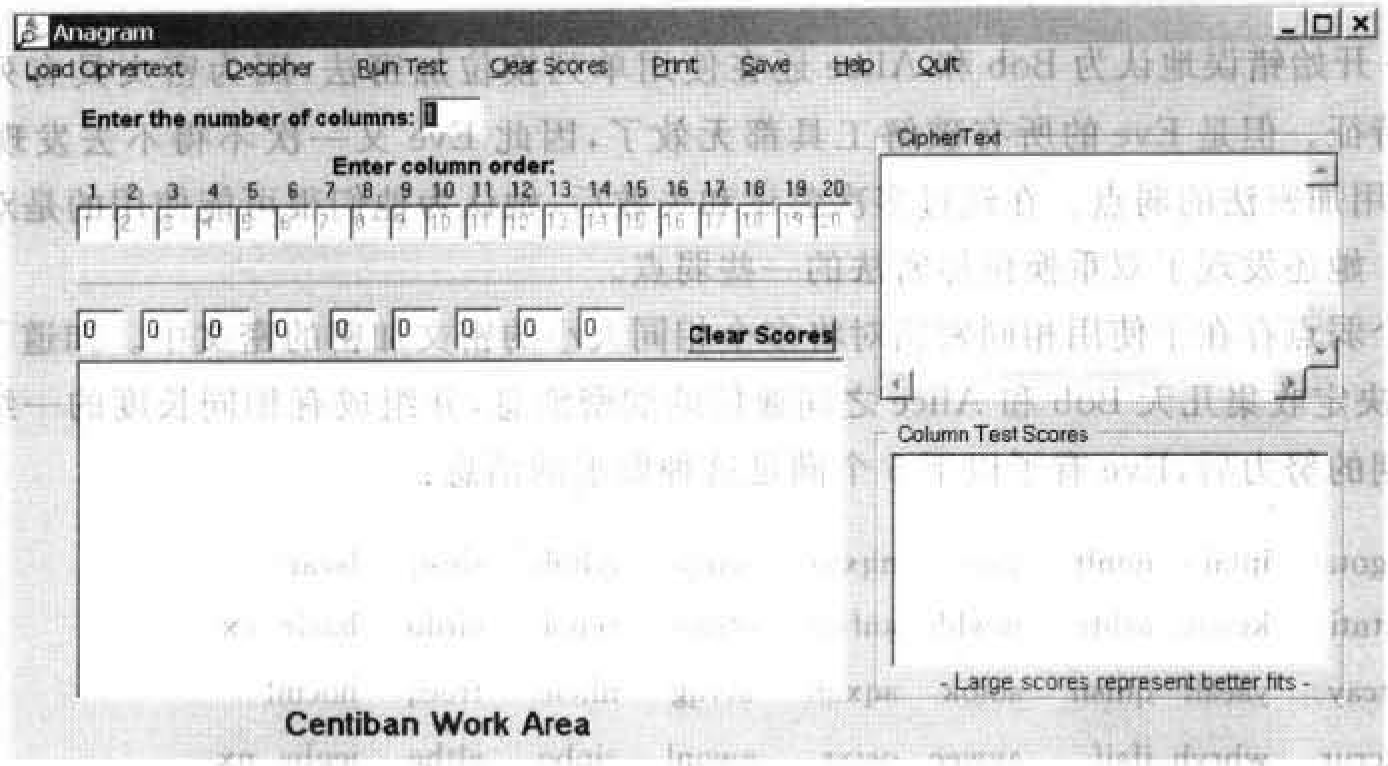


图 5.8 Anagram 窗口

这是 CAP 提供的复杂工具之一。首先输入列数。在工作区中输入列号(以任意顺序)。

5.3 双重换位加密法

Bob 和 Alice 发现单列换位加密法还不够复杂，但他们喜欢使用这种加密法。因此他们进行两次换位，以增加换位的复杂度。他们至少希望这将使 Eve 在尝试破解这种新的换位加密法时，会增加难度。

正如其名字所暗示的那样，双重列换位加密法先用列换位法将明文加密，然后再次用列换位法将第一次换位加密的密文加密。这两次换位所使用的关键词可以相同。经两次换位后，明文字母的位置完全被打乱了。例如，给定明文“Using a cipher twice may improve the strength of the cipher or it may not”。用关键词“what”将其列换位加密后，产生的密文如下：

nircy rttgf ceiyq sahwm mveeh hpomo ugpte iohrt tirtn iceia pesno ehrat

用关键词“next”将上面密文进行第二次列换位加密，产生的密文如下：

irfyh vhoth iianr qnygi amhmp otnis hqcte smeog ittee etqrt cqwep uerrc poaq

正如预先所期望的那样，双重列换位加密法产生的结果是使字母的顺序打得更乱了。Bob 和 Alice 希望复杂度增加后，能使 Eve 原来用来破解单列换位加密法的工具失效。而且让他们高兴的是，CAP 软件能直接实现双重换位加密法，不需要他们手工进行两次单列换位操作。

5.3.2　使用 CAP 软件破解双重换位加密法

Eve 很幸运，她不用手工去破解双重换位加密法，因为 CAP 软件提供了一些工具来破解该加密法。在 Transposition 工具中，有一个 MultiColumn 按钮，单击该按钮将弹出如图 5.9 所示的窗口。要使用这个工具，在主窗口 Ciphertext 中输入密文样本，然后在 Double Column Analysis 窗口中选取 Ciphertext 菜单的 Load Next Ciphertext。它可将每个密文样本排列成行，并定义其列。最后，使用 Centiban Tests 和 Worksheet 将这些列排列成最佳的顺序。

本章只介绍了众多换位加密法中的一些。其实 CAP 软件还实现了其他一些换位加密法。

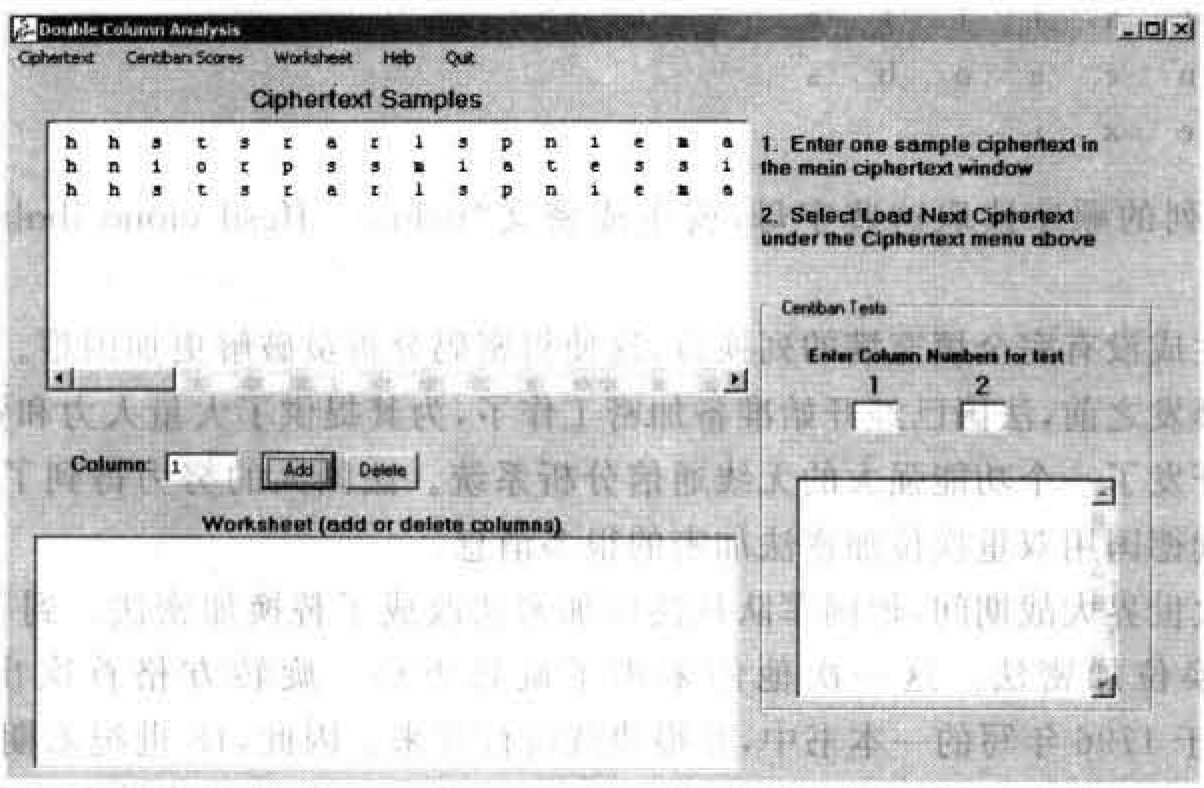


图 5.9　Double Column Analysis 窗口

5.4　换位加密法的历史简介

换位加密法有各种有趣的历史，从希腊人使用的 skytale 加密法到美国国内战争中联军使用的加密法。但换位加密法的使用并没有到此为止。在加密历史中，换位加密法在很长时间中都有其位置。事实上，今天，作为替换—置换加密法（第 7 章将介绍）的一部分，它们仍在使用。

例如，第一次世界大战时期，德国使用过一种著名而复杂的双重换位加密法。首先是将一个关键词短语转换成一个数字序列（使用前面介绍过的字母顺序方法）。例如，如果关键词短语为“next time”，那么其数字短语为如下所示：

n	e	x	t	t	i	m	e
5	1	8	6	7	3	4	2

5.3.1　双重换位加密法分析

Eve 开始错误地认为 Bob 和 Alice 还在使用单列换位加密法，因为密文具有列换位法的所有特征。但是 Eve 的所有破解工具都无效了，因此 Eve 又一次不得不去发现 Bob 和 Alice 所用加密法的弱点。在经过多次尝试和失败后，她认为他们很可能使用的是双重换位加密法。她还发现了双重换位加密法的一些弱点。

一个弱点存在于使用相同密钥对将多个相同大小的密文加密的密文中。知道了这一弱点，Eve 决定收集几天 Bob 和 Alice 之间通信的加密消息，并组成有相同长度的一组。经过一个星期的努力后，Eve 有了以下 5 个满足这种要求的消息：

1.	egota	iptai	qunlt	vsoit	nqxns	seoqb	iyhoh	sicsq	lscar	
	xtnti	kewte	arbtr	mwldi	anfxo	oldao	repoi	eirdu	baalr	cx
2.	ttcay	yaeah	qinsn	aoiok	aqxun	utvqg	nhele	treeq	hoeml	
	xcrur	wbryh	ifaif	aywec	osixr	awonl	cinbo	althe	iegbe	nx
3.	tdhit	htbaa	qewwi	rioon	rqxnm	udeqe	aeset	tahyq	dnsve	
	xiekd	rawrh	eobot	ocher	otuxn	owhfo	maute	kotly	bello	nx
4.	oohoe	ihtmt	qtdnd	daaoe	wqxmt	mxeqo	odaoe	wtaaq	mywbn	
	xcwni	herie	iaaru	euhem	ttvxe	eaeel	rwrte	ftech	ioaew	rx
5.	ocahl	weihb	qtsnn	eaoli	eqxbf	lsaqn	ioofo	hwtoq	onwtu	
	xhtet	tauio	ecbne	bantc	tdaxa	ekoro	aeovo	ofnws	btvue	wx

所有这些都有 92 个字符，如果 Bob 和 Alice 使用的是两个相同的密钥加密这些消息的，Eve 就可以破解该加密法了。Eve 将所有消息对齐叠起排列，也就是说，一个在另一个的上面。这样就组成了一个 92×5 的矩阵，表示一个大的单列换位：

egota	iptai	qunlt	vsoit	nqxns	seoqb	iyhoh	sicsq	lscar	xtnti
kewte	arbtr	mwldi	anfxo	oldao	repoi	eirdu	baalr	cx	
ttcay	yaeah	qinsn	aoiok	aqxun	utvqg	nhele	treeq	hoeml	xcrur
wbryh	ifaif	aywec	osixr	awonl	cento	althe	iegbe	nx	
tdhit	htbaa	qewwi	rioon	rqxnm	udeqe	aeset	tahyq	dnsve	xiekd
rawrh	eobot	ocher	otuxn	owhfo	maute	kotly	bello	nx	
oohoe	ihtmt	qtdnd	daaoe	wqxmt	mxeqo	odaoe	wtaaq	mywbn	xcwni
herie	iaaru	euhem	ttvxe	eaeel	rwrte	ftech	ioaew	rx	
ocahl	weihb	qtsnn	eaoli	eqxbf	lsaqn	ioofo	hwtoq	onwtu	xhtet
tauio	ecbne	bantc	tdaxa	ekoro	aeovo	ofnws	btvue	wx	

现在 Eve 所要做的是重新排列这些列，以便每行都还原到明文形式。这就简化成了一个大的从颠倒字母顺序构成词的问题。

在这 5 个单列换位加密的密文中，显然“q”和“x”是作为空字符来使用的(注意到只有其中几个列才有“q”或“x”)。这些列应移到密文的末尾。剩下其他列的排列可以通过查看先导字母和可能的单词来完成。

将明文“Bob I need to see you at the office now Alice”填写在这个数字序列的下面：

5	1	8	6	7	3	4	2
b	o	b	i	n	e	e	d
t	o	s	e	e	y	o	u
a	t	t	h	e	o	f	f
i	c	e	n	o	w	a	l
i	c	e					

然后将这些列填写在相同的数字序列下面：

5	1	8	6	7	3	4	2
o	o	t	c	c	d	u	f
l	e	y	o	w	e	o	f
a	b	t	a	i	i	i	e
h	n	n	e	e	o	b	s
t	e	e	x	j			

最后，按列的顺序读取这些字母，就生成密文“oebne ffesd eiouo ibola htcoa excwi ejtyt ytne”。

通常会生成没有完全填充满的列换位，这使得密码分析员破解更加困难。但是，在第一次世界大战爆发之前，法国已经开始准备加密工作了，为其提供了大量人力和资源。他们最初的成就是开发了一个功能强大的无线通信分析系统。法国人的努力得到了回报，他们可以破解和阅读德国用双重换位加密法加密的很多消息。

在第一次世界大战期间，德国军队从换位加密法改成了替换加密法。到了 1916 年，他们又回到了换位加密法。这一次他们采用了旋转方格。旋转方格首次出现在 C. F. Hindenburg 于 1796 年写的一本书中，并很快就流行开来。因此，18 世纪末期使用很频繁，后来又在第一次世界大战时期被德国使用。图 5.10 是一个典型的旋转方格和一些样本明文（“this is an example of a turning grille for us”）。如果将旋转方格放在明文上面，那么显示的第一个密文集就如图 5.11 所示。这些密文按行的顺序写为：“hinxetios”。

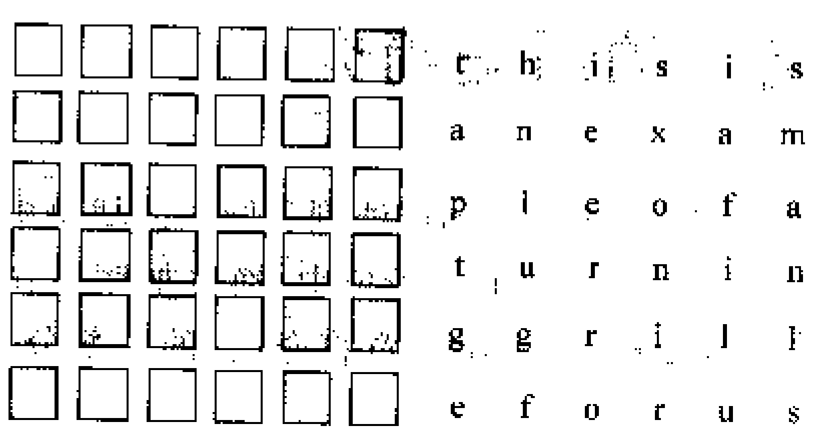


图 5.10 一个旋转方格示例与明文

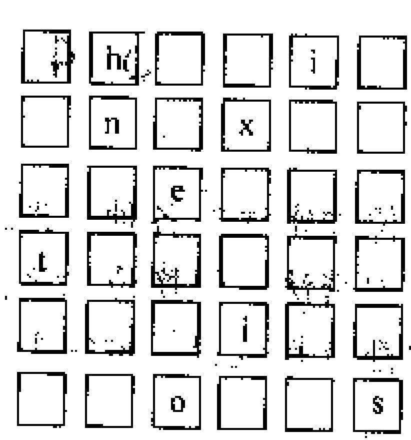


图 5.11 旋转方格的第一个应用

旋转方格按顺时针旋转 90°就显示第二个密文集，如图 5.12 所示。这样密文就是“hinxetios **iampouile**”了。方格再旋转 90°就可显示第三个密文集，如图 5.13 所示，而密文成了“hinxetios iampouile **tseanrlfu**”。最后一步的旋转如图 5.14 所示，而最终的密文为“hinxetios iampouile tseanrlfu salfrnggr”。

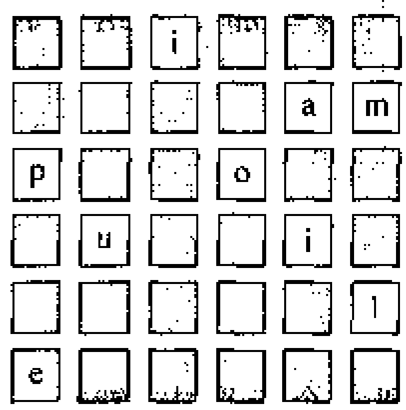


图 5.12　旋转方格第一次旋转

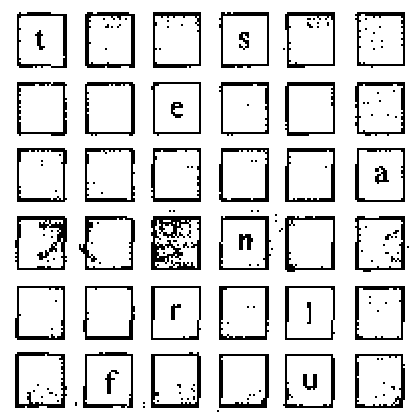


图 5.13　旋转方格第二次旋转

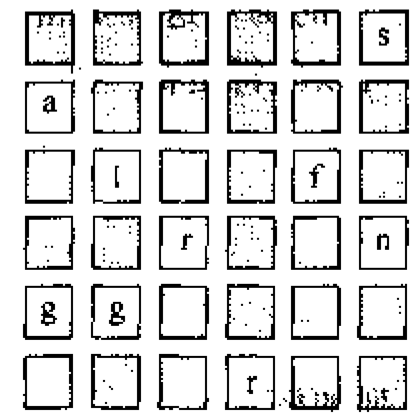


图 5.14　旋转方格最后一次旋转

你也许会认为一战时期法国人是加密大师，但值得注意的是，他们实际上使用的是一个比德国人脆弱的换位算法——称为中断列换位法。在这种加密法中，某些预定的对角线字母先读取，然后再读取各列，读取各列时忽略已读的字母。例如，在下面所示的模式中，首先读取对角线上的字母，然后再读取各列：

4	2	7	1	3	5	6
t	h	i	s	i	s	a
w	e	a	k	c	i	p
h	e	r	i	t	w	a
s	b	r	o	k	e	n

结果密文为“haik aito sk eeb ic twhs swe pan irr”(这里分段显示，实际密文并不如此)。

5.5　本章小结

本章介绍了一种不同的加密法——换位加密法。该加密法不是用一个(或一组字母)替换一个(或一组字母)，而是将明文字母进行重新排列。这使得该加密法的解密就像迷宫游戏一样，需要将所有字母放回到其正确的位置。这种加密法的密钥通常是一个几何图形。明文按某种顺序填写在这个图形中，按另一种不同的顺序读取出来以生成密文。这种新加密法的结构图如图 5.15 所示。

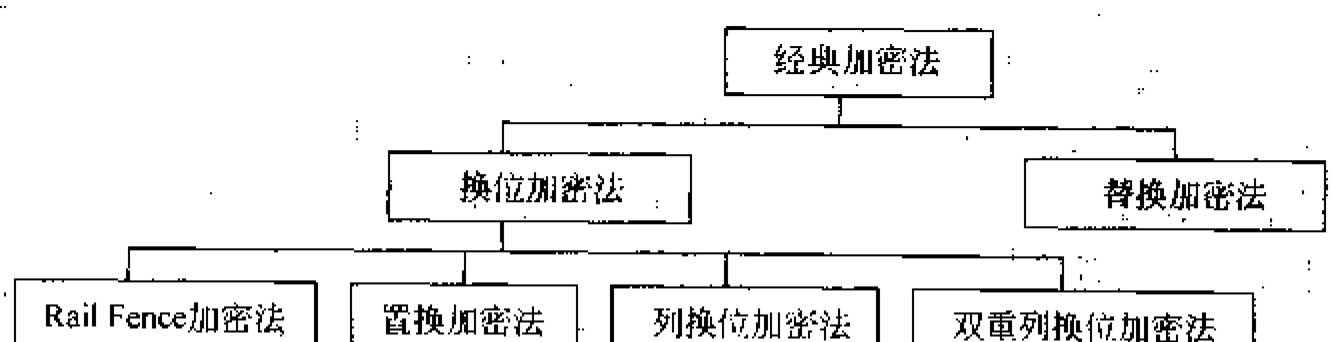


图 5.15　本章介绍的换位加密法

Alice、Bob 和 Eve 之间新一轮的斗争又是难以确定的。Alice 和 Bob 发现了一种保护其消息的新方法，但 Eve 发现了一些破解这些加密法的新工具。尤其是，Eve 发现了同时使用多个密文消息来破解双重换位加密法。这对她后来遇到的更复杂的加密系统非常有用。Alice 和 Bob 知道了，正因为他们实现了两个连续的加密法，他们没有获得任何优势。但

是，他们的确对这种加密法感到了一些希望，如果能找些其他方法，与这些加密法组合起来，或许可以取得更高一级的保护。

5.6 重要术语

列加密法	一种换位加密法，其中，明文按行写入到一个矩形中，然后以一定的顺序读取列以生成密文。
双重列加密法	一种换位加密法，它首先应用列换位加密法生成一个中间密文，然后用另一个换位加密法将中间密文加密成最终密文。
单图换位	一种重新排列字母的换位加密法。
多图换位	一种重新排列单词或字母组的换位加密法。
换位加密法	一种重新排列密文字母顺序的加密法。

习　题

1. 如下密文是使用 rail-fence 换位加密法(4×14)加密生成的：

aellneepiedhcs

请还原成明文。

2. 请填写以下信息：

加密法	操作小结	弱点小结
置换加密法		
列换位加密法		
双重列换位加密法		

3. Bob 和 Alice 有一个想法，认为既然置换加密法可行，为什么不来回进行两次呢？这个主意好吗——也就是说，这比一次置换更安全吗？为什么？

4. 请接着完成第 5.3 节介绍的 Eve 破解换位加密法的工作。

5. 请接着完成第 5.4.1 节介绍的 Eve 破解双重换位加密法的工作。

6. 以下密文是用置换加密法加密而生成的：

eacil dieen whple hitht rieps tmatu iicno ipreh ecsna geotm tetit bonur zozzb

假设我们知道该密码中含有单词“alice”，请还原成明文。

7. 以下密文是用列换位加密法加密而生成的：

anrjc esdhs qenns bcdet eeiok utlri nqlee uabto ihucf
yoqie hssrh nnoio otoqe mliea stwlt ummbq iopnv kited

请还原成明文。

8. Eve 发现 Bob 和 Alice 改换为双重换位加密法了。她截获了以下密文集：

ymqscaqhbgoxewlocrenqrbxlpqvoorktilqufaeqtooriox
qkidiuqseunxllsotidlqyixocqkaioatboqdibiqaccwisx
jryccoqnbutxirdadseuqnbxlwqfesoitiyqoaaoqeotvunx
usanwaqrembxsloioihiqtixoaqnaeaysbgqttbtqacaddhx
saidcdqgbtlxdwucsneiqibxloqdokljeiuqwfayqnoinaix

请还原这些明文。

第二部分　现代加密法

历史上，第一部分中所介绍的加密法的开发和实现大多数是由手工完成的。第一部分中介绍的大多数加密分析技术也是由手工完成的。当然，利用 CAP 软件，计算机现在能完成所有这些工作了。

在第二部分中，我们将正式进入计算机加密的世界。加密法和分析工具不再由手工实现了。整个加密法将充分利用计算机善于处理加密的复杂性的特征。但是，正如你将看到的那样，现代加密法还是扎根于经典加密法的。你在第一部分中所学到的知识，在第二部分中仍很有用，因为这些知识是你学习现代加密法的基础。

第6章 流加密法

6.0 概　　述

在人类历史上，消息的传递是靠人工来完成的。交通工具仅限于步行或骑马。因此，消息到达目的地很慢。如果消息很重要，就要求某些保护措施，可能会进行加密（也是由手工来完成的）。由于传递方式的原因，无论加密与否，重要消息的传递还需要有护卫人员。如果消息被截获，无论护卫人员是回来报告与否，一个不争的事实是，敌人都将获得该消息了。

随着电报技术的发展，消息可以很快地发送到很远的地方，而且不需要送信人了。但是，敌人可能窃听电路，与目标接收方同时接收到消息。如果无保护地发送消息，谁也不知道消息是否被截获了。可以将电报线路保护起来，以尽量减少损失。在这种情况下，消息的安全度降低了。加密作为保护重要信息的一种手段就显得更重要了。

随着无线电的问世，无须电线，消息就可以很快地发送到遥远的地方了。消息的来源地和目的地也更有机动性，可以从一个地方移动到另一个地方仍保持联系。这种变革的代价是使消息完全失去了安全性，敌人随时都可能在"偷听"。加密成了信息保护的关键，它是确保他人偷听到消息但无法理解的惟一方法。

今天，计算机改变了一切。无论信息是什么形式，数量有多大，发送的距离多远，都可能被人共享（或截获）。现在，加密对保护个人隐私和国家安全都是至关重要的。计算机不仅改变了信息的管理方法，也改变了隐藏信息的方法。新的加密法是基于计算机的特征而不是语言结构的了。新加密法设计与使用的焦点放在二进制（位）而不是字母上了。

本节介绍计算机的一些特征，这些特征不仅改变了信息的存储方法，也改变了信息的加密方法。如果你对这些内容熟悉，可以跳过本节，直接学习6.1节。利用信息的计算机表示，将介绍一种基于位流的简单加密法。这里介绍的流加密法虽然体现了计算机的某些特征，但这种加密法的思想并不是新内容。事实上，第3章介绍的自动密钥加密法就很像本章的二进制流加密法。

6.0.1 计算机的特征

计算机技术带来的基本改变是信息的表示方法。在其内部，计算机是以二进制位（0和1）来表示信息的。因此，字母不再是信息存储的方法，而是现今二进制位。所有信息都必须转换成计算机的位，以进行存储和操作，人需要阅读时，又得将这些位转换回字母和数字的形式。

在二进制中，数字容易表示。以二进制计算类似于十进制计算，只是二进制只有两个数字(0 和 1)而已。二进制数与十进制数的比较如下：

二进制	十进制	二进制	十进制
0	0	110	6
1	1	111	7
10	2	1000	8
11	3	1001	9
100	4	1010	10
101	5		

使用二进制数字系统来表示十进制数字时，所生成的数字串更长。但是，所有可运用于十进制数的数学运算(加、减等)，也可运用于二进制数。因此，从十进制到二进制的改变并不大。

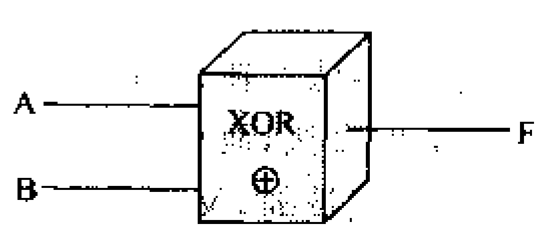
人们需要计算机是可以同时表示字符和数字的。这并不是听起来这么容易，因为字母与二进制数串之间并没有数学关系，需要定义某种关系以避免混乱。这种关系在计算机系统开发的早期就已经定义了，称为 ASCII 码(American Standard Code for Information Interchange，美国信息交换标准码)。它还定义了一些特殊字符，如换行或回车。

例如，计算机内部以 8 个二进制位 01100001 来表示字母“a”。在 CAP 软件中，选取 Plaintext 菜单的 to Binary 菜单项，即可将明文转换成相应的 ASCII 码。

由于从字母转换成 ASCII 码对用户是透明的(用户在计算机屏幕或打印机上所看到的是字母，而不是二进制位)，因此仍可以编写计算机代码，在字母一级上执行替换和换位加密法。CAP 软件一直就是这么做的。但是，由于计算机是以 0 和 1 的字符串来表示明文的，因此可以设计基于二进制字符串而不是单个字母的加密法。例如，字母“a”的 ASCII 码位 01000001。如果加密法将某些 1 改变为 0，将某些 0 改变为 1，就可以将表示字母“a”的二进制字符串变成 00101101，而这是“+”的 ASCII 码。要开发这种加密法的问题是：“如果随机地将 1 变成 0，将 0 变成 1，以后如何还原成原始的二进制字符串?”答案是使用二进制位的一个特殊运算，即异或逻辑运算(XOR)。

6.0.2 XOR 逻辑运算

XOR 是一个特殊的二进制逻辑运算，有两个输入，一个输出。如果两个输入不同，则输出为 1。图 6.1 总结了 XOR 逻辑运算的真值表，列出了所有可能的输入组合和输出结果。



A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

图 6.1　XOR 逻辑运算及其真值表

该逻辑可用于二进制串的"随机"改变位运算。假设 XOR 的一个输入是明文的二进制位流，另一个输入是随机密钥二进制位串，那么其输出是密文二进制位流。例如，对以 ASCII 码表示的单词"The"，可以如下进行加密：

明文	T	h	e
明文的二进制位	01010100	01101000	01100101
密钥的二进制位	00001101	00110100	01101101
密文的二进制位	01011001	01011100	00001000
密文	Y	\	"SI"shift in

上面示例首先需要注意的是，密文不仅仅是字母，可能是特殊或不可打印的字符。这是一个有两种不同字母的加密示例：一种用于明文，另一种用于密文。后面将详细介绍这种两种字母的情况。对于这种加密法，问题是如何从密文还原明文"The"。答案是："将密钥的二进制位与密文的二进制位进行 XOR 逻辑运算"。这很简单，具体如下所示：

密文	Y	\	"SI"
密文的二进制位	01011001	01011100	00001000
密钥的二进制位	00001101	00110100	01101101
明文的二进制位	01010100	01101000	01100101
明文	T	h	e

因此，位一级的加密法可以利用 XOR 逻辑运算和随机密钥流来构造。这就是下一节将要介绍的内容。

6.1 流加密法

一个简单的流加密法需要一个"随机"的二进制位流作为密钥。通过将明文与这个随机的密钥流进行 XOR 逻辑运算，就可生成密文。将密文与相同的随机密钥流进行 XOR 逻辑运算即可还原明文。该过程如图 6.2 所示。

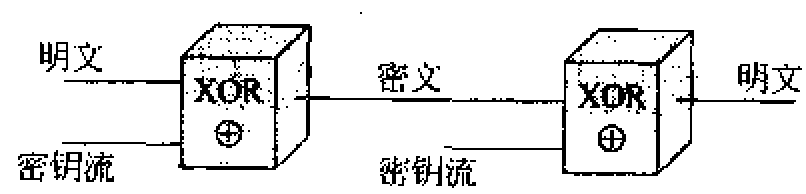


图 6.2 流加密法

要实现 XOR 逻辑运算很简单，当作用于位一级上时，这是一个快速而有效的加密法。惟一的问题是必须解决如何生成随机密钥流。这之所以是一个问题，是因为密钥流必须是随机出现的，并且合法用户可很容易再生该密钥流。如果密钥流是重复的位序列，容易被记住，但不很安全。而一个与明文一样长的随机位序列很难记住。因此，这是一个两难的处境，如何生成一个"随机"位序列作为密钥流，要求易于使用，但又不能太短以至于不安全。这非常类似于第 3 章中介绍的 Vigenere 字母密钥加密法。在字母密钥加密法中，密钥流是

从明文或密文中形成的。在这两种情况下,密钥流都是以一个关键词为开始。解决流加密法的密钥也是这样：开发一个随机位生成器,它是基于一个短的密钥来产生密钥流的。生成器用来产生密钥流,而用户只需记住如何启动生成器即可。

有两种常用的密钥流生成器：同步与自同步的。同步生成器所生成的密钥流与明文流无关。因此,如果在传输时丢失了一个密文字符,密文与密钥流将不能对齐。要正确还原明文,密钥流必须再次同步。自同步流加密法是根据前 n 个密钥字符来生成密钥流的。如果某个密文字符有错,在 n 个密文字符后,密钥流可自行同步。

有多种产生同步密钥流生成器的方法。最普遍的是使用一种称为线性反馈移位寄存器(linear feedback shift register, LFSR)的硬件设备。

6.1.1 线性反馈移位寄存器

在开始介绍流加密密钥设计之前,我们需要先介绍一下有关硬件电路的知识。这没什么奇怪的,因为我们正在介绍的加密法是基于计算机,而不是靠手工操作字母的。

一个非常有用的硬件设备是移位寄存器。该设备可以保存一个二进制位集,这些位可以以两种方法加载到寄存器中：一种是并行加载,即,如果寄存器可保存 8 位,则 8 位同时加载；另一种是使用移位寄存器来加载。典型的 8 位寄存器一次加载一位,如图 6.3 所示。

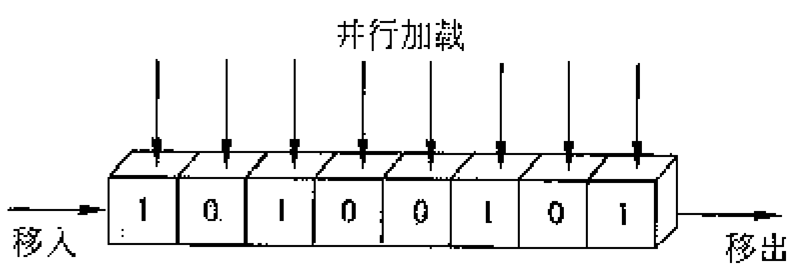


图 6.3 一个 8 位移位寄存器

在移位方式中,每个往右移,最右边的位被丢弃,而最左边的被输入位替代。对于如图 6.3 所示的移位寄存器,如果移入位为 0,那么,移位操作后,寄存器的内容如图 6.4 所示。

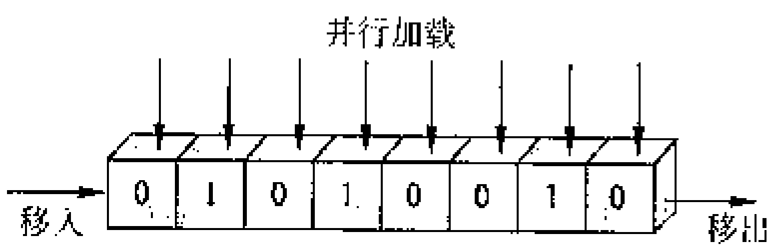


图 6.4 移入一个 0 以后的寄存器内容

只有在移入端有一个移入位,移位寄存器就会在移出端移出一位。该设备本身不能生成一个长的作为密钥的"随机"位流。毕竟,它只是在移出端输出所移入的内容。因此,如何生成一个随机的移入位序列呢？这就需要用到密钥生成器的反馈部分了。选取移位寄存器的一些单元,将它们进行 XOR 逻辑运算后,再将结果作为输入,这就是说,将当前寄存器中的内容进行逻辑运算,再反馈给输入。

假设我们有如图 6.5 所示的线性反馈移位寄存器(LFSR),其中,单元 2 和 5 的内容进行 XOR 逻辑运算后作为移入内容。LFSR 的运行结果如表 6.1 所示。

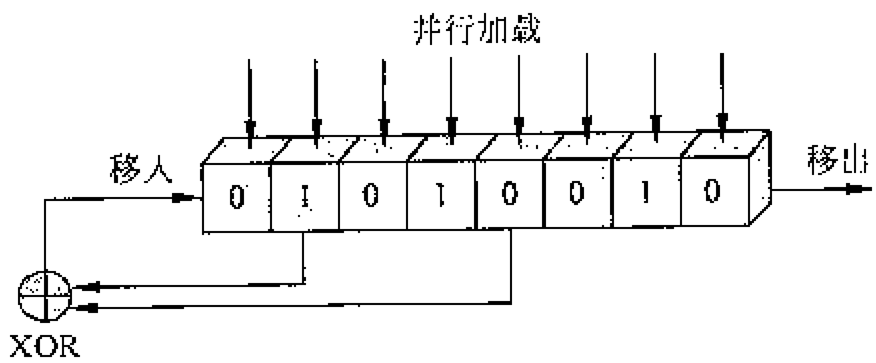


图 6.5 一个带反馈的移位寄存器

表 6.1 LFSR 的运行示例

移入	单元 1	单元 2	单元 3	单元 4	单元 5	单元 6	单元 7	单元 8	移出
1	1	1	0	1	0	0	1	1	1
0	1	1	1	0	1	0	0	1	1
1	0	1	1	1	0	1	0	0	0
1	1	0	1	1	1	0	1	0	0
1	1	1	0	1	1	1	0	1	1
0	1	1	1	0	1	1	1	0	0

移出流就是密钥(这里是 0 1 0 0 1 1)。一个 LFSR 可以生成足够长的密钥的二进制位以匹配明文的二进制位。要解密密文,只需运行具有相同初始内容的 LFSR 即可。最终,各单元的内容将循环回开始值,密钥流就重复了。设计这种用于生成随机位的 LFSR 的一个问题是,对于一个给定的 LFSR,可生成最长的无重复序列位是多少。这个问题在设计一个加密法时特别重要,因为这牵涉到该加密法的破解难易程度,具体内容将在后面章节中介绍。显然,如果 LFSR 使用的是一个 n 位的寄存器,寄存器的内容不会重复,除非这 n 个组合的所有可能情况都出现过了。因此随机数的周期为 2^n-1(减 1 是因为我们从不会使所有寄存器的内容都为 0,本章后面的问题将讨论这种情况)。实际周期取决于寄存器的初始内容以及 LFSR 所使用的反馈模式。某些设置可以保证周期最大。周期最大的 LFSR 的结构可根据寄存器的不同长度来分类。例如,如果单元 5 和 6 做 XOR 逻辑运算后的结果作为反馈位的 6 位的 LFSR 周期最大。

6.1.2 LFSR 周期分析

常见的 LFSR 可以用存储的位和移入位的函数来表示。对于图 6.6 所示的 LFSR,其函数为:

$$b_n = c_1b_1 \text{XOR} c_2b_2 \text{ XOR} \cdots \text{XOR} c_nb_n$$

其中,如果第 i 位选做 XOR 操作,那么 $c_i=1$;否则为 0。

每个 LFSR 有一个关联的特征多项式,它与决定下一个移入位的函数不同。该多项式与 LFSR 的数学结构相关。对图 6.6 所示的示例,特征多项式为:

$$p(x) = c_nx^n + c_{n-1}x^{n-1} + \cdots + c_1x + 1$$

与前面一样，其中，如果第 i 位选做 XOR 操作，那么 $c_i=1$；否则为 0。事实证明，如果 LFSR 的特征多项式是“基本的”(有时候也称为“不可约分的”)，那么 LFSR 就有一个最大的序列。如果不能约分了，且将 $p(x)$分为 x^k-1 的最小整数 k 为 2^n-1，那么多项式的次数 n、$p(x)$是基本的。

如图 6.7 所示的是一个有 3 个元素的 LFSR，它的特征多项式为：

$$p(x) = x^3 + x + 1$$

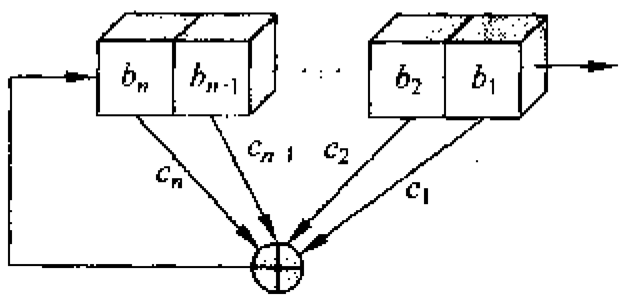


图 6.6 一个常见的 LFSR

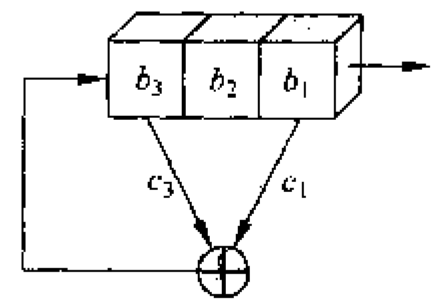


图 6.7 有 3 个元素的 LFSR

它满足基本多项式的要求。因此图 6.7 的 LFSR 具有如图 6.8 所示的最大周期。

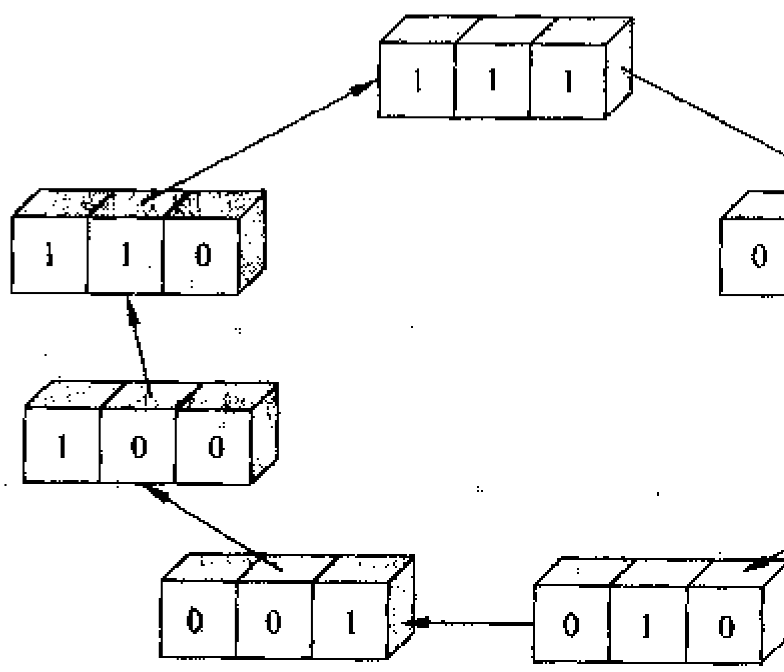


图 6.8 最大周期

而图 6.9 所示的 LFSR 则没有最大周期。这是一个有 4 个元素的 LFSR，其特征多项式为：

$$p(x) = x^4 + x^3 + x^2 + x + 1$$

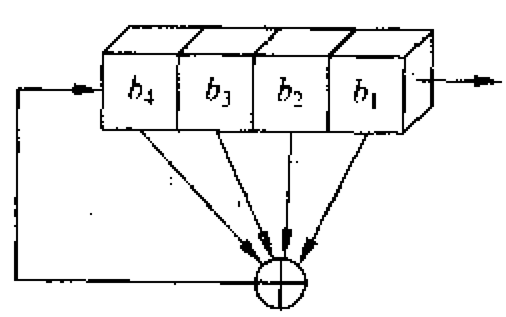


图 6.9 有 4 个元素的 LFSR

它的周期如图 6.10 所示。如果该 LFSR 初始为该周期中的一个位模式，那么它将总是生成该周期中的随机位串。

对 LFSR 的基本特征多项式，运用数学知识可以确定它是否有最大周期。对反馈函数

使用一个非 XOR 逻辑运算，可以知道，有 4 个元素的 LFSR，要使其具有最大周期，那么应用第 3 和 4 位作为反馈；而有 8 个元素的 LFSR，要使其具有最大周期，那么应用第 8、6、5 和 4 位作为反馈。

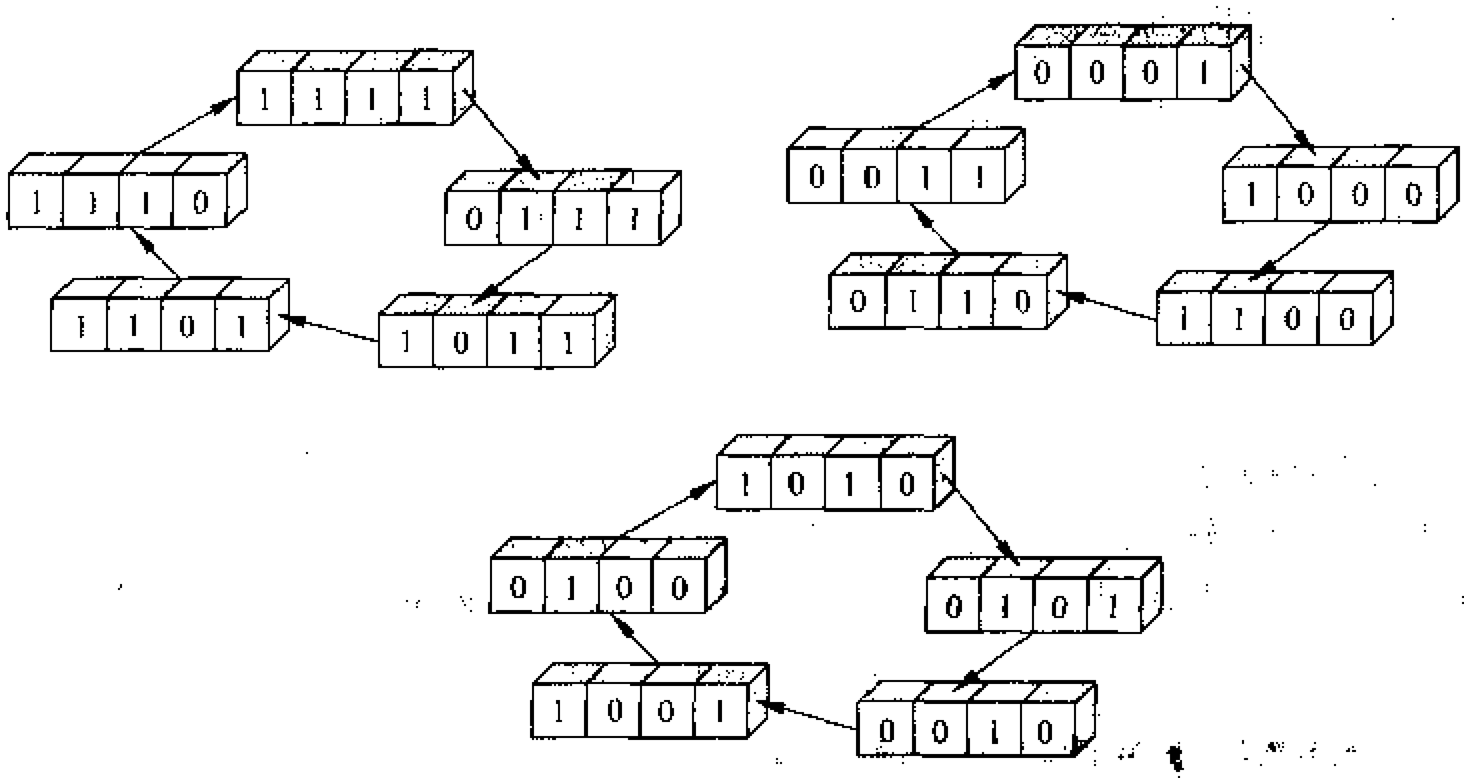


图 6.10 非最大周期

6.1.3 随机位测试

设计随机数字生成器的另一个问题是，如何保证是随机的位？如果这些位不完全是随机的，那么攻击者就有可能发现这些位的某种模式，从而可能发现密钥的全部或部分。为了避免这个问题，开发了几种测试随机性的不同方法。这些测试可用于确定某个随机生成器是否能安全使用。

要从数学的角度来证明某个随机位生成器是否真的产生了随机位流，这是不可能的。所能做的是看看随机位流是否具有所期望的特征。因此，有多组标准化的随机性测试方法。FIPS 140-1 测试组包含有一些很明显的测试，也有一些不太明显的测试。该测试组可以接收 20 000 个二进制位。第一个测试是**单个位测试**(monobit test)，校验 1 和 0 的个数是否大致相等。我们期望在随机位流中 1 和 0 的个数大致相等。这个测试过程是计算 1 的个数：如果其个数在 9654～10 346 之间，那么这个位流就通过了单个位测试。

但是，通过了单个位测试并不能保证该位流就真的是随机的。FIPS 140-1 的另一个测试是**扑克牌测试**(poker test)。在扑克牌测试中，将这 20 000 个二进制位分成若干段，每段有 4 个二进制位。每段所表示的十进制数是 0～15。对于一个真正随机的位流，0～15 的个数也应是随机分布的。假设 n_i 是数字 i 的个数，即 n_1 是 0001 段的个数 n_8 是 1000 段的个数。将这些值代入下式：

$$X = \frac{16}{5000}\sum_{i=0}^{15} n_i^2 - 5000$$

如果 $1.03 < X < 57.4$，测试通过。

第三种随机性测试法称为**连串测试**(runs test)。连串就是指 1 或 0 连续的序列。在真

正的随机位流中,连串的长度应是随机分布的。例如,在 0110111101 中,一个连串的长度为 2(即第 2～3 位的 11),另一个连串的长度为 4(即第 5～8 位的 1111),此外,还有 4 个长度为 1 的连串。要测试 20 000 个二进制位,先计算各种长度的连串的个数。如果每个连串的数量位于以下区间之内,那么该位流就通过了此测试:

连串长度	1	2267～2733
	2	1079～1421
	3	502～748
	4	223～402
	5	90～223
	6+	90～223

美国国家标准与技术协会(NIST)也给出了一些随机性测试的建议。FIPS 140-1 的一些就在其中。CAP 软件提供了这些测试的一部分,可以用来评估流加密法或其他的位加密法。要运行这些测试法,从 CAP 的主菜单中选取 Random 菜单,然后再选取 Test 子菜单,将弹出如图 6.11 所示的 Randomness Tests 窗口。

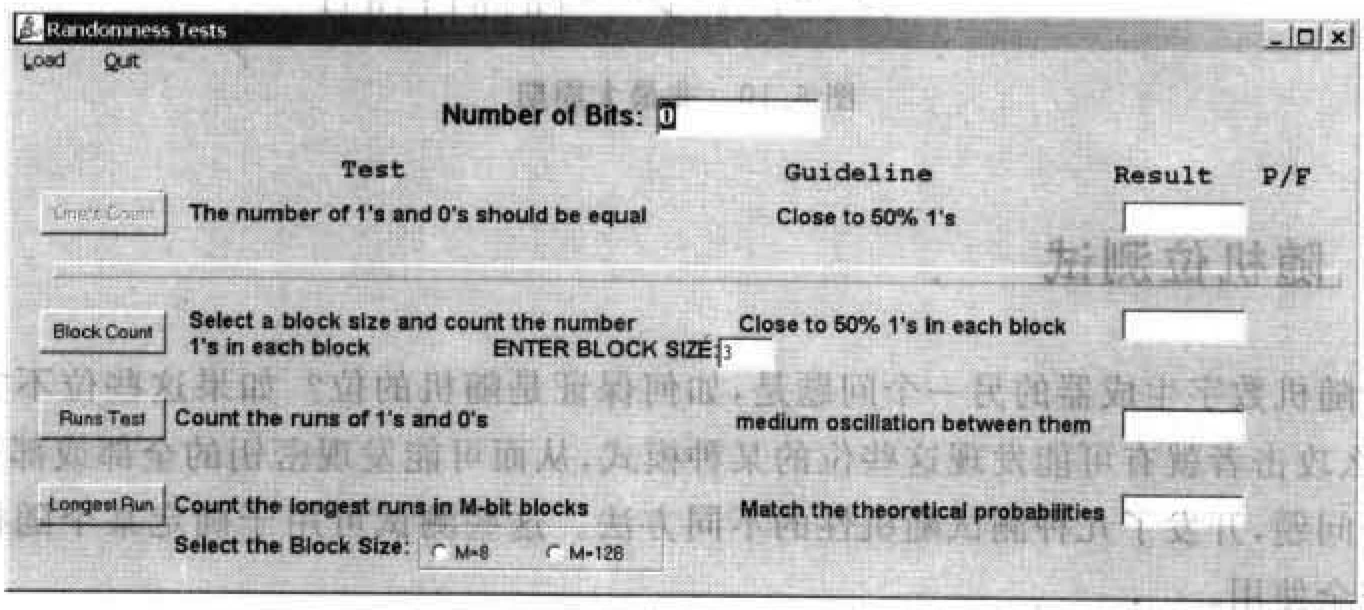


图 6.11 Randomness Tests 窗口

一个可以接受的随机位序列至少应通过 CAP 软件提供的所有测试。

6.1.4 在 CAP 软件中实现流加密法

流加密法不能用手工来实现。它需要将明文用二进制位表示,并生成一个长的随机位流。它生成的密文字符也是以二进制位表示的。Alice 和 Bob 很幸运,CAP 软件提供了流加密法。从 Cipher 主菜单中选取 Stream Cipher 选项,将弹出如图 6.12 所示的窗口。

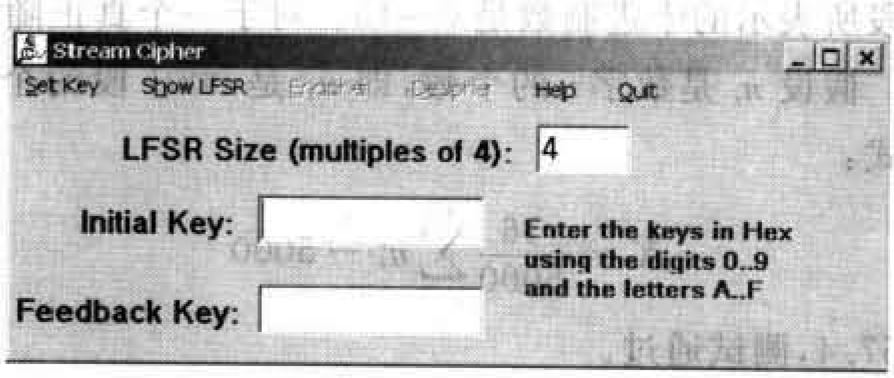


图 6.12 Stream Cipher 窗口

CAP 提供的流加密法也是基于一个 LFSR 的。你需要输入移位寄存器的大小，它必须是 4 的倍数。然后，输入移位寄存器的初始值（十六进制的数）。反馈密钥决定哪些寄存器的单元将参与 XOR 逻辑运算以生成输出位。这也必须是十六进制的数。CAP 软件还有另一个特征，如果你对你的随机密钥生成器没有把握，这个特性很有用。一旦指定了寄存器的大小和这两个密钥，就可以从主菜单中选取 Show LFSR，弹出如图 6.13 所示的 LFSR Structure 窗口，显示 LFSR 的内容和反馈链接。

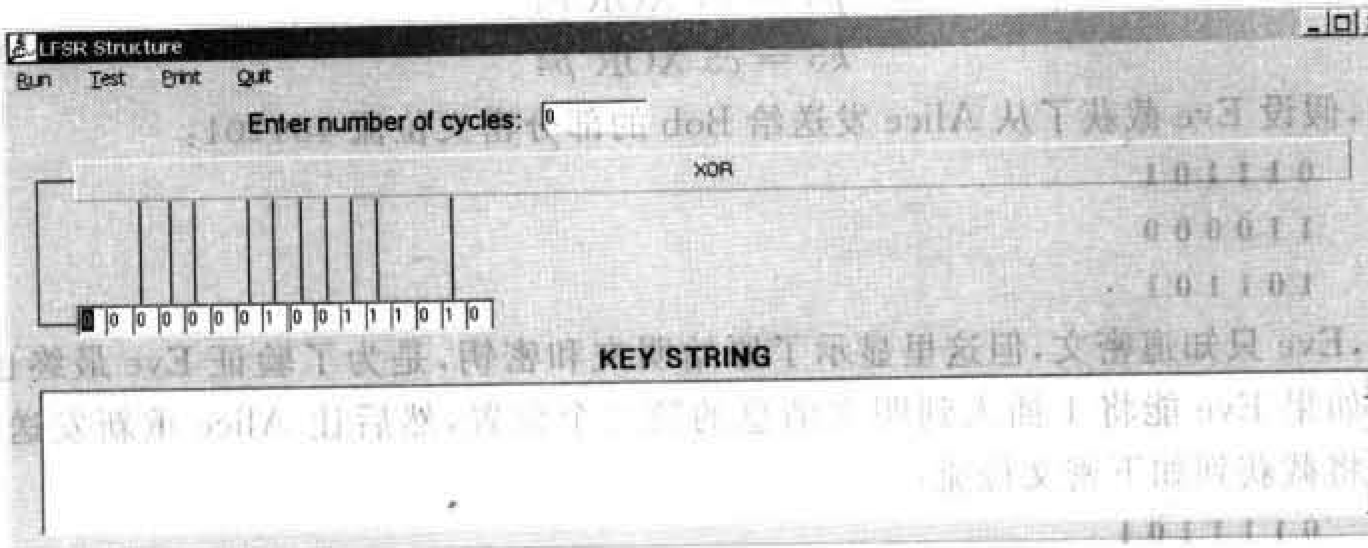


图 6.13 LFSR Structure 窗口

从这个窗口中，通过选取 Test 菜单项，就可以真正运行一些简单的随机性测试了。你还可以选取 Run 菜单项来看看你的 LFSR 周期。

6.2 破解流加密法

Alice 和 Bob 认为他们现在已经进入了 21 世纪，他们终于找到了一种可以保护他们通信的方法，可以逃过 Eve 的破解了。但不幸的是，Eve 还是很快就跟上了他们的步伐。她发现 Alice 和 Bob 使用的是流加密法，因此开始查找这种加密法的构成方法。她发现了三种类型的攻击法，即插入攻击法、位串匹配攻击法和已知明文攻击法。

6.2.1 用插入攻击法破解流加密法

这种攻击法很简单，但要求能在明文流中插入一个位，并截获密文流。假设原始明文、密钥流和密文为：

p1 p2 p3 p4 p5...

*k*1 *k*2 *k*3 *k*4 *k*5...

*c*1 *c*2 *c*3 *c*4 *c*5...

如果 Eve 能在明文流中插入一个已知位 *p*，例如，插入在第一个位后面，然后用同样的密钥加密后发送，结果将是：

p1 p p2 p3 p4 p5...

*k*1 *k*2 *k*3 *k*4 *k*5 *k*6...

*c*1 *c*2 *c*3 *c*4 *c*5 *c*6...

由于 Eve 知道插入的已知位和这两个密文流，所以就可以构建一个等式组，从而可以求解出密钥和实际的明文。第一个求解出来的密钥位是 $k2$，因为 $k2=c2$ XOR p，知道了 $k2$ Eve 就可以利用 $p2=k2$ XOR $c2$ 得出原来的第二位 $p2$，知道了 $p2$，Eve 就可以利用 $k3=c3$ XOR $p2$ 得出 $k3$。以此类推，就可以有以下方程组：

$$p3 = k3 \text{ XOR } c3$$
$$k4 = c4 \text{ XOR } p3$$
$$p4 = k4 \text{ XOR } c4$$
$$k5 = c5 \text{ XOR } p4$$

例如，假设 Eve 截获了从 Alice 发送给 Bob 的部分密文位流 101101：

明文　**0 1 1 1 0 1**
密钥　**1 1 0 0 0 0**
密文　**1 0 1 1 0 1**

当然，Eve 只知道密文，但这里显示了原始明文和密钥，是为了验证 Eve 最终也能得出这些值。如果 Eve 能将 1 插入到明文消息的第二个位置，然后让 Alice 重新发送该消息。那么 Eve 将截获到如下密文位流：

明文　**0 1 1 1 1 0 1**
密钥　**1 1 0 0 0 0 1**
密文　**1 0 1 1 1 0 0**

有了这两个密文流，并知道第二位为 1，Eve 就可以如下求出原始密钥和明文：

$$k2 = 0 \text{ XOR } 1 = 1$$
$$k3 = 1 \text{ XOR } 1 = 0$$

$$p2 = 1 \text{ XOR } 0 = 1$$
$$p3 = 1 \text{ XOR } 0 = 1$$

这样攻击法的关键是得有办法在明文的某个位置插入一个位，并使发送方重新发送一次。在实际中，不太可能有这种情况，但是，这说明了流加密法存在一个弱点，值得 Alice 和 Bob 关注一下。现在，Eve 决定尝试另一种更容易破解流加密法的方法——可能词攻击法（probable-word attack）。

6.2.2 可能词攻击法一：位串匹配攻击法

可能词攻击有两种方法，其中之一是数学特性更浓一些。Eve 决定首先采用这种方法，因为它是更一般的解决方法。要用已知明文攻击法破解基于 LFSR 的流加密法，Eve 必须获得明文的位串及其相应的密文串，以及 LFSR 的大小（如果不知道 LFSR 大小，可以尝试几个合理的值，这个方法仍然可行。）。要重构密钥生成器，只需要找出反馈位。

Eve 首先从已知的 m 位明文串 $p_i(i=0,1,\cdots,n-1)$ 以及其相应的 m 位密文串 $c_i(i=0,1,\cdots,n-1)$ 着手。利用下式，Eve 可以得出密钥串 k_i：

$$k_i = p_i \text{ XOR } c_i$$

通常，对于 LFSR，密钥串的各位由下式给出：

$$k_{m+j} = \sum_{i=0}^{m-1} a_i k_{i+j} \bmod 2$$

其中，和与 2 的模就相当于一个 XOR 逻辑运算；m 是 LFSR 的大小；a_i 表示参与反馈的位，也就是说，如果 $a_4=1$，那么该位就反馈回 LFSR。

如果已知明文流的位数 n 等于或大于 $2m$（m 是 LFSR 的大小），那么就可以通过求解如下线性方程得出 a_i：

$$(a_0,a_1,\cdots,a_{m-1})=(k_{m+1}k_{m+2},\cdots,k_{2m})\begin{bmatrix}k_1 & k_2 & \cdots & k_m\\ k_2 & k_3 & \cdots & k_{m+1}\\ \vdots & \vdots & & \vdots\\ k_m & k_{m+1} & \cdots & k_{2m-1}\end{bmatrix}^{-1}$$

一旦知道了 a_i，那么就知道了 LFSR 的结构，也就可以得出密钥了。

例如，如果已知明文—密文位流，通过将这两个位流做 XOR 逻辑运算即可：

明文	0110 0001 0110 1100
密文	1011 0100 0001 0011
密钥流	1101 0101 0111 1111

假设是一个 8 位的移位寄存器（因此可知 $m=8$），可以从以下矩阵得出反馈链：

$$\begin{bmatrix}1&1&0&1&0&1&0&1\\1&0&1&0&1&0&1&0\\0&1&0&1&0&1&0&1\\1&0&1&0&1&0&1&1\\0&1&0&1&0&1&1&1\\1&0&1&0&1&1&1&1\\0&1&0&1&1&1&1&1\\1&0&1&1&1&1&1&1\end{bmatrix}$$

求解该矩阵的逆：

$$\begin{bmatrix}1&1&0&1&0&1&0&1\\1&0&1&0&1&0&1&0\\0&1&0&1&0&1&0&1\\1&0&1&0&1&0&1&1\\0&1&0&1&0&1&1&1\\1&0&1&0&1&1&1&1\\0&1&0&1&1&1&1&1\\1&0&1&1&1&1&1&1\end{bmatrix}^{-1}=\begin{bmatrix}1&0&1&0&0&0&0&0\\0&1&1&0&0&0&0&1\\1&1&0&0&0&0&1&0\\0&0&0&0&0&1&0&1\\0&0&0&0&1&0&1&0\\0&0&0&1&0&1&0&0\\0&0&1&0&1&0&0&0\\0&1&0&1&0&0&0&0\end{bmatrix}$$

将密钥位（k_9，…，k_{16}）与矩阵的逆相乘：

$$(11000000)\begin{bmatrix}1&0&1&0&0&0&0&0\\0&1&1&0&0&0&0&1\\1&1&0&0&0&0&1&0\\0&0&0&0&0&1&0&1\\0&0&0&0&1&0&1&0\\0&0&0&1&0&1&0&0\\0&0&1&0&1&0&0&0\\0&1&0&1&0&0&0&0\end{bmatrix}=(01111111)$$

可得到反馈向量为(11000000),于是可知该 LFSR 有 8 位,其中第 0 和第 1 位做 XOR 逻辑运算后反馈回寄存器。

6.2.3 可能词攻击法二:单词匹配攻击法

事实证明,知道了密文的某个可能的单词,再加上一些聪明的操作,Eve 就可以破解流加密法。单词匹配攻击法与前面介绍的位串匹配攻击法不同在于,不知道相应的密文。Eve 必须在密文流中寻找与可能词匹配的位置。为了清楚地说明这个攻击法的过程,我们假设 Alice 和 Bob 使用的是一个简单形式的 LFSR,其中,只有第一个位和第二个未知位用作反馈。尽管事实并不是如此,当记住,这里介绍的方法对复杂的 LFSR 同样适用。

假设 Alice 和 Bob 使用一个 6 位的 LFSR,其中把位 1 和 4 的逻辑运算结果作为输入,如图 6.14 所示。

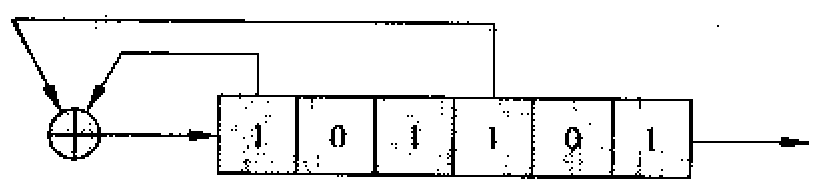


图 6.14 一个 6 位的 LFSR

Eve 截获了以下加密位流:

111000011111100110001110010001101011100000100011

她猜测第一个单词为"The"。如果她的猜测没错,那么将"The"的 ASCII 码与密文流做 XOR 逻辑运算,就可以得出密钥的前一部分的位。

T h e

01010100 01101000 01100101

11100001 11111001 10001110 010001101011100000100011

10110101 10010001 11101011

这只能得出密钥位的字母为"e"。要找出密钥的其他字母,需要找出用于 XOR 逻辑运算以生成 LFSR 输入位的两个寄存器位。通过将密钥流中的位对做 XOR 逻辑运算以生成一个新的位流即可:

初始密钥 101101011001000111101011

1 (把第一对 1 和 0 做 XOR 逻辑运算)

初始密钥 101101011001000111101011

11 (把第二对 0 和 1 做 XOR 逻辑运算)

以此类推,最后可以得出:

初始密钥 101101011001000111101011

11011110101100100011110

查看原始密钥和原始密钥两两做 XOR 逻辑运算后的结果,可以发现一个长的相等位

模式。上面示例的相等位模式为：

初始密钥　101101011001000111101011

　　　　　110111101011001000111110

这些位模式的差距就确定了 LFSR 中进行 XOR 逻辑运算的第二个位。这上面的示例中，其差距为 4，这说明 Alice 和 Bob 所使用的 LFSR 是以位 1 和 4 作为反馈来生成密钥流的。有了这些信息，通过将当前密钥与密钥位 4 以后的位做 XOR 逻辑运算，Eve 就可以重构整个密钥流。结果如下：

密文流　111000011111100110001110010001101011100000100011

密钥流　101101011001000111101011001000111101011001000111

明文流　010101000110100001100101011001010110111001100100

　　　　T　h　e　e　n　d

如果不能产生可读的结果，那么单词“The”的位置是不正确的，Eve 必须将该单词移位并重复上面的过程。

6.2.4　使用 CAP 软件破解流加密法

CAP 软件为 LFSR 流加密法的加密与攻击提供了必要的工具。例如，如果 Eve 截获了 Alice 和 Bob 之间的通信消息，她怀疑这些消息经过了 LFSR 流加密法加密（但她不知道 Alice 和 Bob 所使用的移位寄存器的大小、移位寄存器的初始值，或反馈链），她仍可以破解该加密法。就在 Alice 和 Bob 以为他们最终难倒了 Eve 时，Eve 已经确信消息是以单词“Alice”打头的了，因为该消息是从 Bob 发送给 Alice 的。有了这个信息，Eve 就可以破解该加密法和 LFSR 了。

6.3　其他流加密法的实现

所有具有密钥流与明文流相互作用的加密法就是流加密法。密钥流不一定非得是由单个 LFSR 生成的，也不一定非得只是二进制位。还有其他实现流加密法的方法，它们为密钥生成随机位流的方法各不相同。

人们设计了其他一些生成随机位流的方法，以提供密钥的随机性，如使用多个 LFSR。事实上，可以使用多个不同的 LFSR，其中，一个 LFSR 的输出是基于另一个 LFSR 随机选取的。这就称为**非线性生成器**（nonlinear generator）。有一种用来执行这个选择功能的硬件设备，称为**复用器**（multiplexer，MUX）。MUX 就是一个具有多个输入和一个输出的数字设备。其目的是从这多个输入中选取一个作为输出。如图 6.15 所示，它的操作像一个开关。地址位决定该开关的位置，而它又决定了哪个输入与输出相连。改变地址位就将改变输出连接。

复用器的大小不一。图 6.15 所示的是一个 4-1 的复用器，因为它有 4 个数据输入一个输出。它要求两个地址位，以便决定选取这 4 个数据输入。例如，如果地址位为 00，那么数

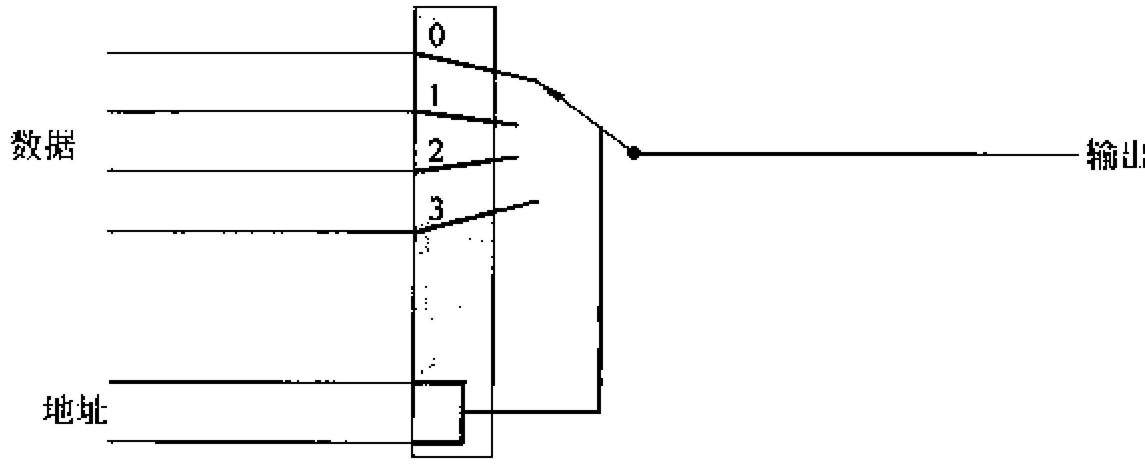


图 6.15 一个 4-1 的复用器

据输入 0 将与输出连接，而其他 3 个数据输入被阻塞。8-1 复用器是一个更大的设备，它具有 8 个数据输入和 3 个地址位。

如果与多个 LFSR 一起使用，复用器就可用于生成更好的随机位密钥流。例如，如图 6.16 所示的系统就使用了一个 4-1 复用器和 5 个不同的 LFSR。

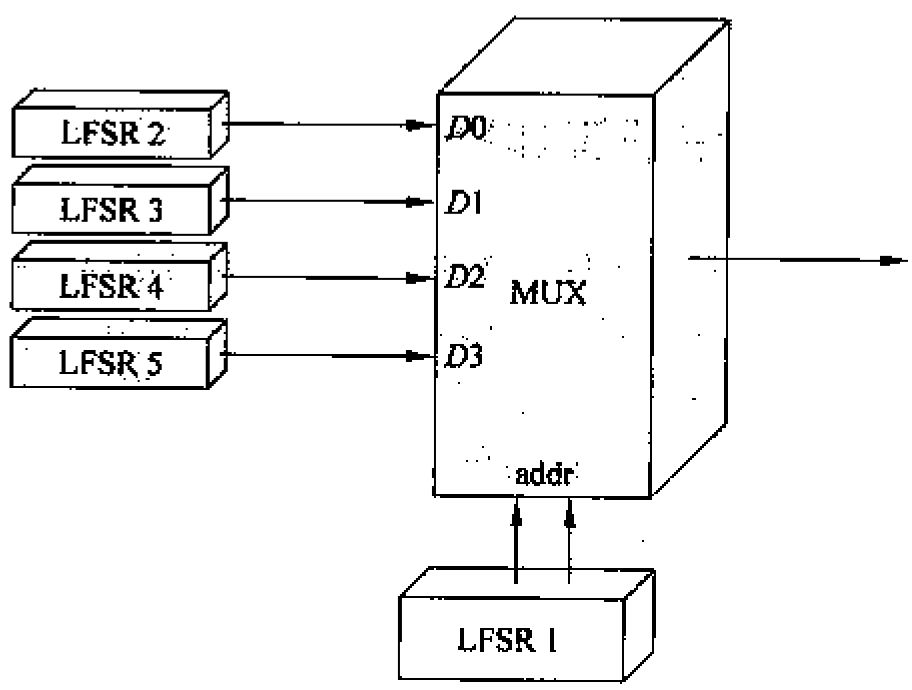


图 6.16 具有多个 LFSR 的生成器

为了给信号加密，很多卫星电视广播公司就使用了这种随机位流生成器。电视信号的每条线路是切断的，由随机生成器的输出决定旋转到某个位置。

6.3.1 RC4

RC4 是由麻省理工学院的 Ron Rivest 开发的。Ron Rivest 是 RSA 的开发者之一。有关 RSA 将在后面介绍。RC4 可能是世界上使用最广泛的流加密法。它已应用于 Microsoft Windows、Lotus Notes 和其他软件应用程序中。它使用于安全套接字层(secure sockets layer，SSL)以保护因特网的信息流。它还应用于无线系统以保护无线链接的安全。RC4 的一个优点是它在软件中很容易实现。

RC4 的大小根据参数 n 的值而变化。RC4 可以实现一个秘密的内部状态，对 n 位数，有 $N=2^n$ 种可能。通常 $n=8$，这也是本书示例所选取值。RC4 可以生成总共有 256($=2^8$)个元素的数组 S。RC4 的每个输出都是数组 S 中的一个随机元素。要实现这些，需要两个

处理过程：一个是**密钥调度算法**(key-scheduling algorithm，KSA)，用来设置 S 的初始排列；一个是**伪随机生成算法**(pseudo random-generation algorithm，PRGA)，用来选取随机元素并修改 S 的原始排序顺序。

KSA 开始初始化 S，即 $S(i)=i$(其中 $i=0\sim255$)。通过选取一系列数字，并加载到密钥数组 $K(0)\sim K(255)$。你不用去选取这 256 个数，只要不断重复直到 K 被填满。数组 S 可以利用以下程序来实现随机化：

```
j:=0;
for i:=0 to 255 do begin
    j:=i+S(i)+K(i)(mod 256)
    swap (S(i),S(j))
end
```

一旦 KSA 完成了 S 的初始随机化，PRGA 就将接手工作，它为密钥流选取字节，即从 S 中选取随机元素，并修改 S 以便下一次选取。选取过程取决于索引 i 和 j，这两个索引值都是从 0 开始的。下面程序就是选取密钥流的每个字节：

```
        i:=i+1 (mod 256)
        j:=j+S(i) (mod 256)
        swap (S(i),S(j))
    t:=S(i)+S(i)(mod 256)
k:=S(t)
```

Alice 和 Bob 对 RC4 感兴趣，所以尝试使用 3 位(从 0～7)的 RC4，其所有操作是对 8 取模(而不是对 256 取模)。数组 S 只有 8 个元素，初始化为：

S	0	1	2	3	4	5	6	7
	0	1	2	3	4	5	6	7

Alice 和 Bob 选取一个密钥，该密钥是由 0～7 的数以任意顺序组成的。例如，他们选取 5、6 和 7 作为密钥。该密钥如下填入密钥数组中：

K	5	6	7	5	6	7	5	6
	0	1	2	3	4	5	6	7

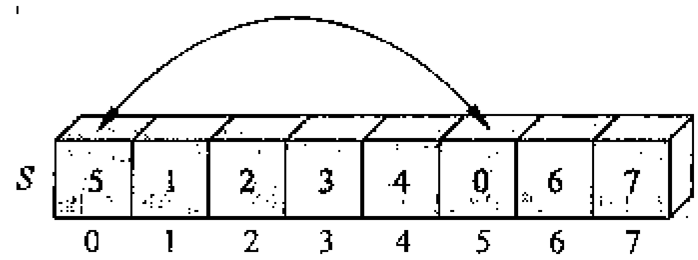
然后利用如下的循环构建实际的 S 数组：

```
j=0;
for i=0 to 7 do
    j=(j+S(i)+K(i)) mod 8;
    swap (S(i),S(j));
```

该循环以 $j=0$ 和 $i=0$ 开始。使用更新公式后 j 为：

$$\begin{aligned} j &= [0+S(0)+K(0)] \bmod 8 \\ &= (0+0+5) \bmod 8 \\ &= 5 \end{aligned}$$

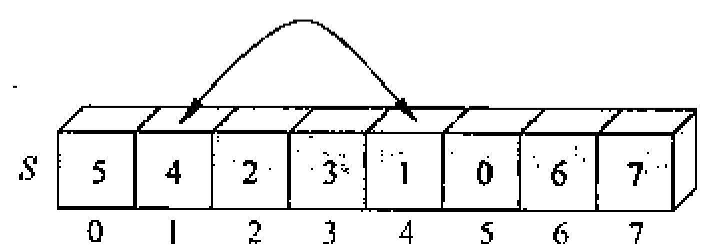
因此，S 数组的第一个操作是将 $S(0)$ 与 $S(5)$ 互换：



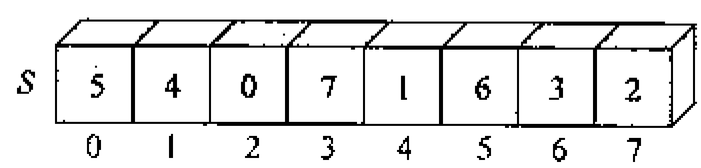
索引 i 加 1 后，j 的下一个值为：

$$j = [5 + S(1) + K(1)] \bmod 8$$
$$= (5 + 1 + 6) \bmod 8$$
$$= 4$$

即是将 S 数组的 $S(0)$ 与 $S(5)$ 互换：



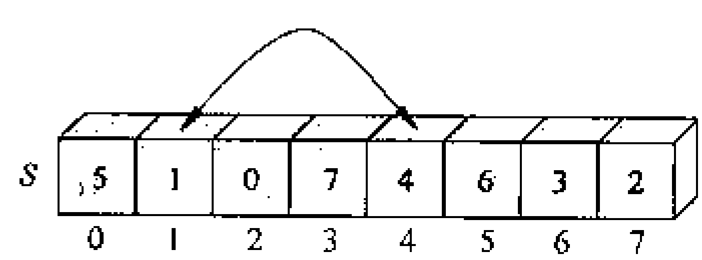
当该循环执行完后，数组 S 就被随机化为：



这样数组 S 就可以用来生成随机数序列了。从 $j=0$ 和 $i=0$ 开始，RC4 如下计算第一个随机数：

$$i = (i + 1) \bmod 8 = (0 + 1) \bmod 8 = 1$$
$$j = [(j + S(i))] \bmod 8 = [(0 + S(1))] \bmod 8 = (0 + 4) \bmod 8 = 4$$

swap $S(1)$ and $S(4)$



然后如下计算 t 和 k：

$$t = [S(j) + S(i)] \bmod 8 = [S(4) + S(1)] \bmod 8 = (1 + 4) \bmod 8 = 5$$
$$k = S(t) = S(5) = 6$$

第一个随机数为 6，其二进制表示为 110。反复进行该过程，直到生成的二进制位等于明文的位。

CAP 软件提供了 RC4 的完整 256 位实现。要使用它，只需在 Ciphers 菜单中选取 RC4 选项即可。

6.3.2 评估 RC4

常见的 RC4 实现是基于 $n=8$ 的(数字为 0～256)。这种系统的初始密钥是 0～256 的整数,共有 256! 即 2^{1600} 种可能。这相当于使用了一个 1600 位的密钥,使强力攻击法变得不可能。

正如前面 Alice、Bob 和 Eve 反复的斗争一样,穷尽密钥搜索法通常不是破解加密法的惟一方法。在 RC4 中,有一些弱点可用来破解该加密法。例如,KSA 永不会生成某类密钥,例如 $j=i+1$ 与 $S(j)=1$。事实证明,这类密钥的数量是所有可能密钥数的 2^{-2n}。当 $n=8$ 时,就是(256! $/2^{16}$)。

由于这些是不可能的,去除这些后,尽管可以减少强力攻击法需要查找的量,但还是不可行的。另一种强力攻击法是专门设计一个高效实现该算法的硬件。这种硬件能加速测试可能密钥的过程。这种设备于 1999 年由 Kundarewich、Wilton 和 Hu 设计出来了。他们设计了一个可编程逻辑设备,在 159 天内就可找出 40 位 RC4 的密钥。这看上去是一个花费了很长的时间,但足以对这种加密法构成威胁了。

6.3.3 A5

A5 流加密法是一种无规律的多 LFSR 系统。如图 6.17 所示,它由三个不同大小的 LFSR 组成,分别为 A、B 和 C。A 有 19 位,B 有 22 位,C 有 23 位。流输出是这些寄存器的 XOR 逻辑运算结果。这些寄存器是无规律调度的,这意味着每个寄存器在不同调度规划中所移的位不同。每个寄存器的时钟信号由 x、y 和 z 的值决定,其中 x 是第一个寄存器的第 9 位,y 是第二个寄存器的第 11 位,z 是第三个寄存器的第 11 位。移位函数实现了三个输

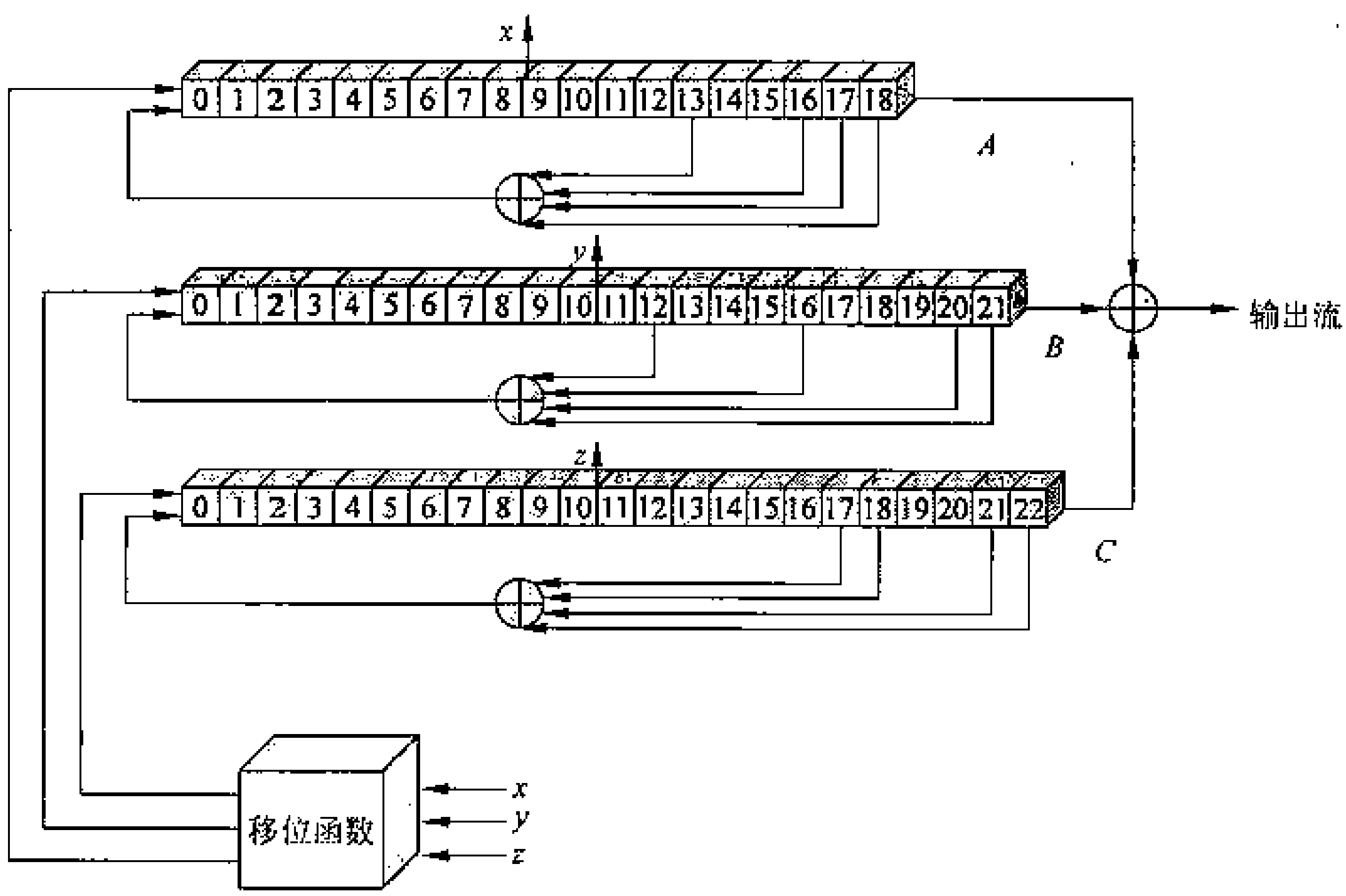


图 6.17 A5 流加密法

入 x、y 和 z 的多数功能。当且仅当两个或多个输入为 1 时，多数函数(maj(x,y,z))为 1。因此，只有 x XOR maj(x,y,z)=0 时 A 才移位；只有 y XOR maj(x,y,z)=0 时 B 才移位；最后，只有 z XOR maj(x,y,z)=0 时 C 才移位。

密钥是这三个寄存器的初始值，这意味着 A5 有一个 64 位的密钥。密钥的大小一定程度上会使人误解，以为密钥越长就越安全，其实不一定，因为已知明文攻击法只需要知道了其中两个寄存器的初始值，就可以计算出另一个寄存器的初始值。这只需要 2^{40} 步就可以得出寄存器 A 和 B。A5/2 是 A5 的一个简单版本，只需要 2^{16} 步就可以破解它。除了有效密钥长度不大外，A5 还有一个冲突问题，即不同的寄存器初始值可能产生相同的密钥流。事实上，研究表明，在寄存器所有初始值中，实际上只有 70%的值可以产生惟一的密钥流。

A5 流加密法已应用于 GSM 通信系统中，用来保护语音通信。GSM 是世界上最大的移动电话系统，它有 2 亿多用户。在美国，GSM 是数字(个人通信服务)网络的一部分。一个 GSM 语音消息被转换成一系列的帧，每帧具有 228 位。每帧用 A5 进行加密，如图 6.18 所示。

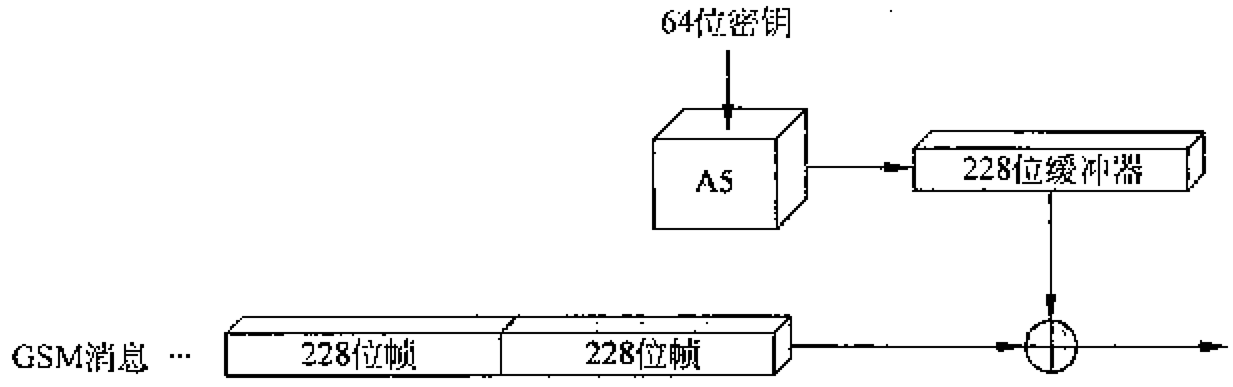
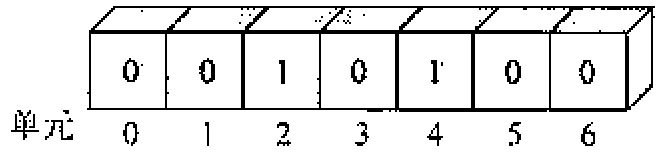


图 6.18 GSM 中使用 A5 流加密法

6.3.4 单元自动操作

一个**单元自动操作**(cellular automation,CA)就是单元的数组(可以是一维或二维的)。二进制 CA 中的单元有两种状态：0 或 1。经过每个时钟滴答后，数组单元的状态将根据相邻单元的状态而发生变化。这种结构首先由 John von Neumann 于 20 世纪 40 年代引入。20 世纪 70 年代早期，当 John Conway 发明了基于 CA 的生命游戏后，人们对 CA 的兴趣开始增加了。从那以后，被用在了很多的应用中，其中之一就是随机位的生成。

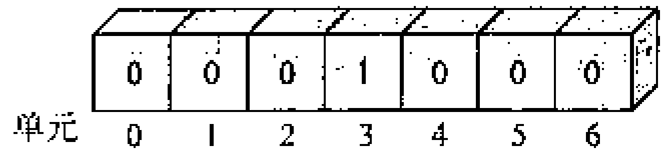
一个简单的 CA 是具有 7 个单元的一维数组，如下所示：



上面的 CA 初始化为 0010100。每个单元根据某种规则改变其状态。一种规则是单元的状态根据其相邻单元的状态而变，也就是说，某个单元的更新值取决于该单元的当前值及其左右单元的值。可以假设 CA 被连成为一个环，因此单元 0 左边是单元 6，单元 6 的右边是单元 0。这种规则可以如下表示：

相邻情况	000	001	010	011	100	101	110	111
新状态	0	0	0	1	0	1	1	1

将这个规则应用到最初的 CA 中，将生成如下的 CA：



用于这种类型的 CA 的规则，可以通过将新状态的位转换成十进制来标识。例如，应用前面的规则后的新状态为 00010111，即为十进制数 23。规则 90 是 01011010。

选取好一个规则、CA 的大小、初始值和提供随机位的单元后，就可以用 CA 来生成随机位。例如，一个有 11 个单元的 CA，其初始值为 01100011000，应用规则 30，第 7 位用作随机位，如图 6.19 所示。

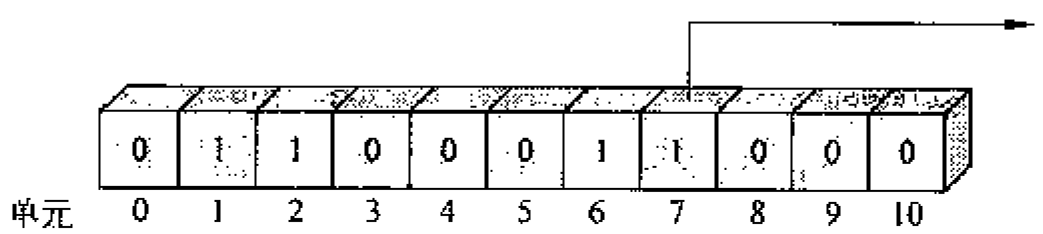


图 6.19 CA 随机位生成器

规则 30 为：

相邻情况	000	001	010	011	100	101	110	111
新状态	0	0	0	1	1	1	1	0

将该规则应用到上面的 CA 示例中，前 4 步生成的随机位如图 6.20 所示。

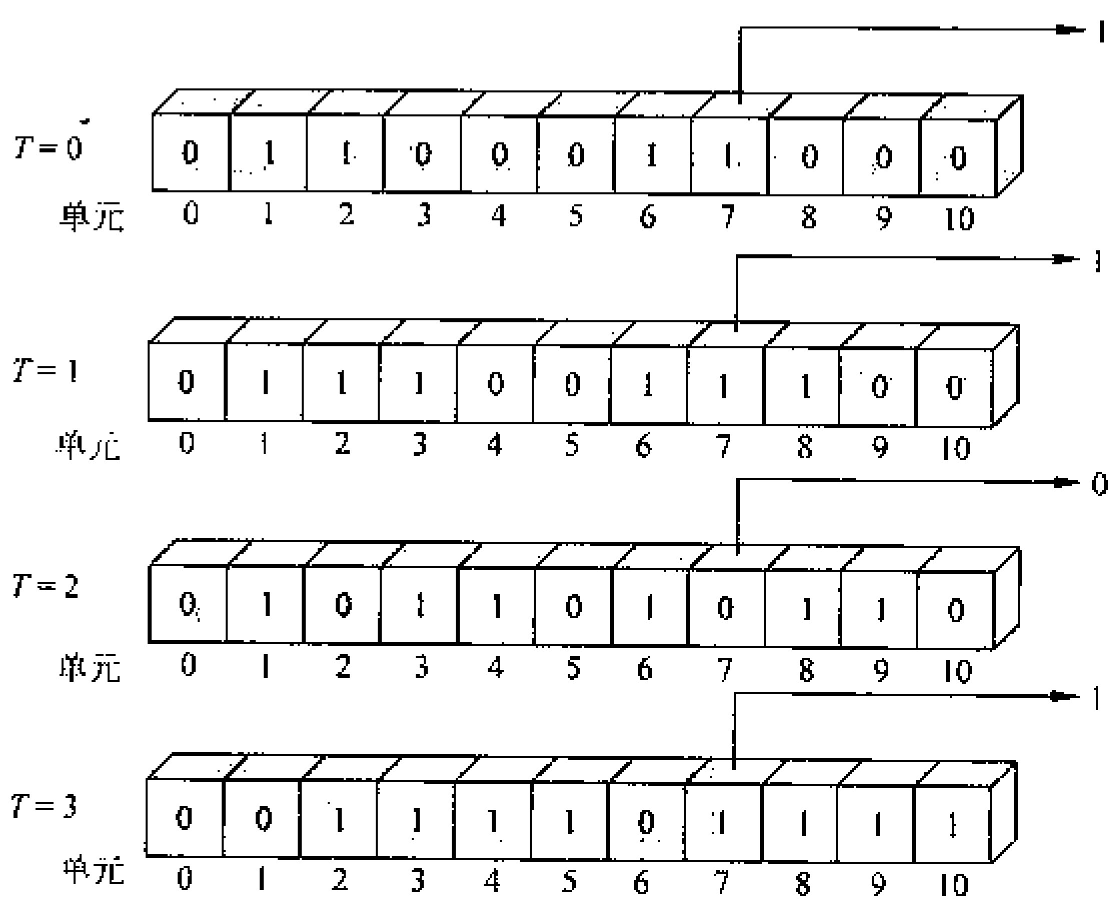


图 6.20 前 4 步生成随机位的情况

与其他的随机数生成器一样，CA 需要解决的问题是："何时会重新重复"，以及"最好的规则是什么"。单元自动操作领域的开拓者 Steven Wolfram，研究了一维 CA 的特征发现，

使用规则 30 并且初始值中只有一个单元的值为非零的情况下，生成最长的周期。

CAP 软件提供了一个小型的一维 CA 系统，用来测试这种随机数生成器的可能性。CAP 的 Cellular Automata 窗口如图 6.21 所示。要在 CAP 中设置一维 CA，首先选取 CA 的长度(输入一个 1～10 的整数)，然后为每个单元输入一个规则，并选定一个单元作为随机位流。随机位流可以用来加解密示范文本，或者用来生成一个随机位流，以作为测试之用。这只需从 Random 菜单项中选取一个选项即可。

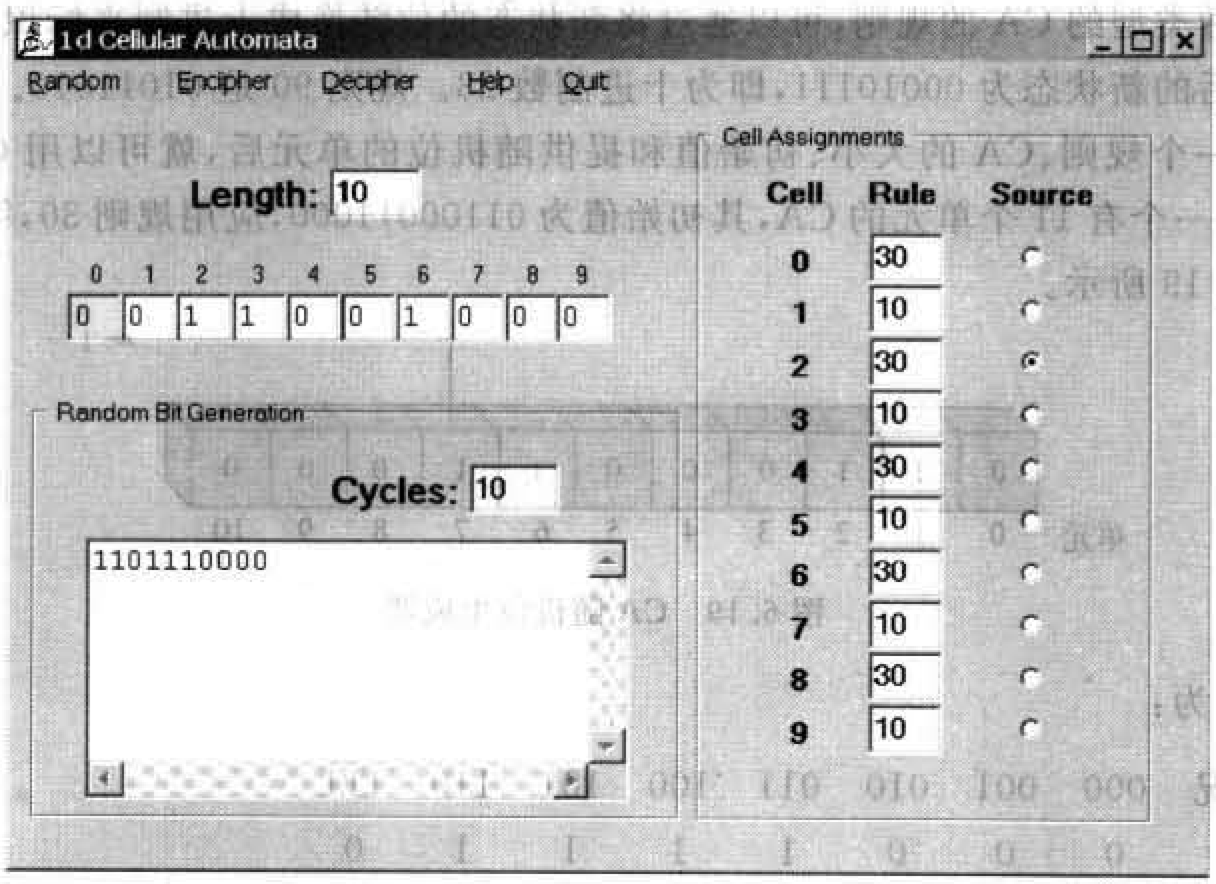


图 6.21 CAP 的 Cellular Automata 窗口

二维 CA 提供了一个功能更强大的随机数生成器，但代价是复杂性增加了。二维 CA 就是一维 CA 的数组，如图 6.22 所示。与在一维 CA 中一样，每个单元的更新值取决于它的当前值及相邻单元的值。在二维 CA 中，相邻单元有两种不同的定义。正如图 6.22 所示的那样，von Neumann 相邻单元包括该单元上、下、左、右的单元。而 Moore 相邻单元则包括该单元四周的 8 个单元。

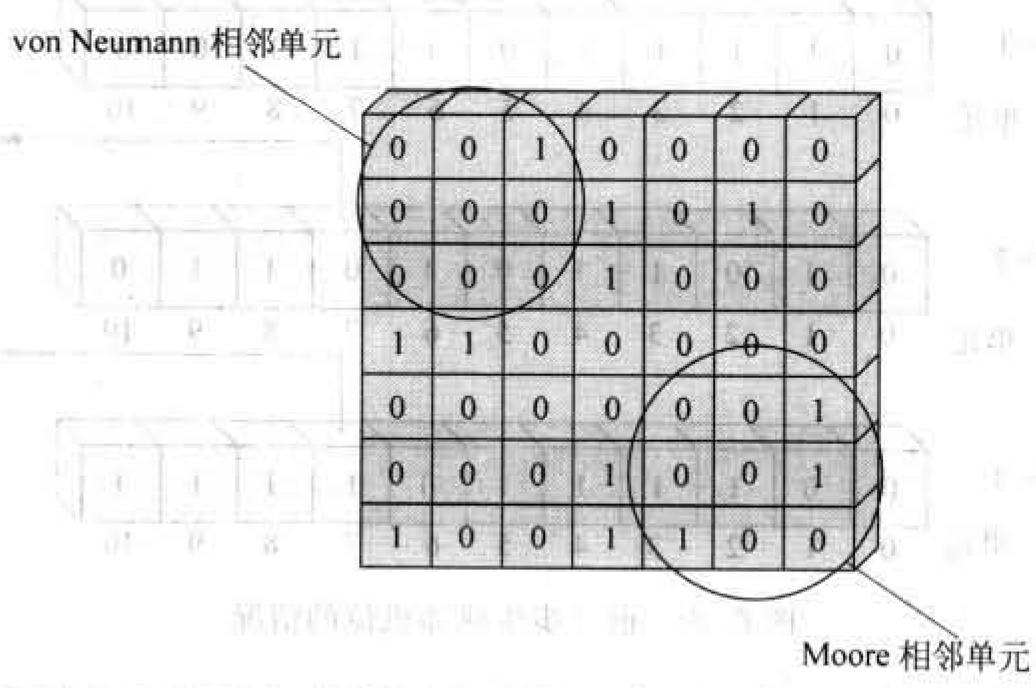


图 6.22 一个二维 CA

二维 CA 的规则很像一维 CA 的。这些规则定义了某个单元是如何根据其相邻单元的值来更新其值的。对于 von Neumann 相邻单元，一个常见的规则定义如下：

$$s_{i,j}(t+1) = X\ \mathrm{XOR}[(C \times s_{i,j}(t))]\ \mathrm{XOR}\ [(N \times s_{i-1,j}(t))]\ \mathrm{XOR}\ [(W \times s_{i,j-1}(t))]$$
$$\mathrm{XOR}[(S \times s_{i+1,j}(t))]\ \mathrm{XOR}\ [(E \times s_{i,j+1}(t))]$$

其中 X、C、N、W、S、E 是取值为 0 和 1 的变量。如果 X 为 1，表示是非线性规则；否则就是线性规则。C、N、W、S 和 E 分别表示中间、北边、西边、南边和东边的单元。这 5 个变量的值参与决定中间单元的更新值。例如，如果 N 为 1，那么北边单元用来决定中间单元的更新值。这 6 个变量的值用来标识每种可能的规则。例如，如果 $(X,C,N,W,S,E)=(001110)$，那么该规则就定义为规则 14，因为 1110 等于十进制的 14。可以用如下公式表示这些规则：

$$s_{i,j}(t+1) = s_{i-1,j}(t)\ \mathrm{XOR}\ s_{i,j-1}(t)\ \mathrm{XOR}\ s_{i+1,j}(t)$$

由于有 6 个变量，所以有 64 种可能的规则。Tommassini 等发现，规则 31、47 和 63 在 8×8的 CA 中可生成更好的随机数。

给每个单元指定一个规则，将 CA 初始化为一个随机状态，就可以生成随机位流了。例如，图 6.23 显示的是一个 3×3 的 CA，其中每个单元指定应用规则 14。中间单元的值可用来生成随机位流。每个单元的初始化如图 6.23(a)所示。经过 3 个步骤后，生成的随机位流为 0110。

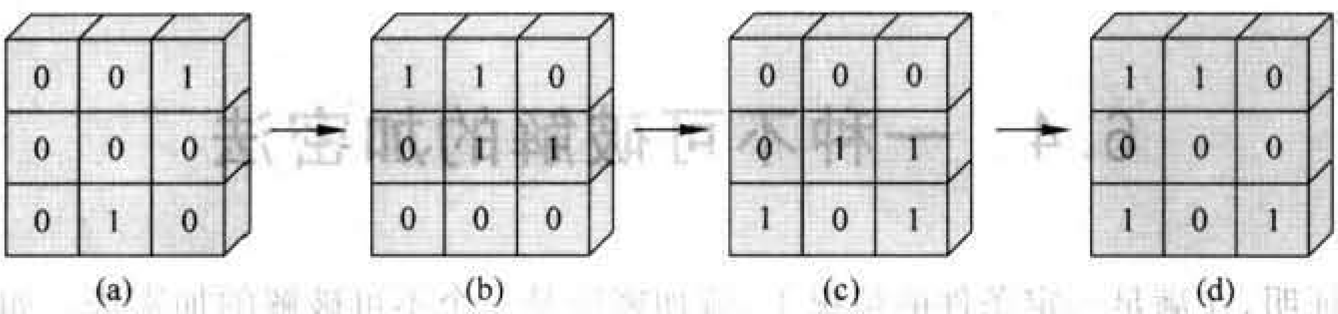


图 6.23 一个 3×3 的 CA

CAP 软件提供了一个简单的二维 CA，以供测试之用。从 Cipher 菜单中选取 2d Cellular Automata 菜单项，将弹出如图 6.24 所示的窗口。CAP 将生成一个 7×7 的 CA。可以为每个单元指定一个规则，或者使用 Cell Rule Assignments 面板的快捷键为所有单元指定相同的规则。与一维 CA 一样，二维 CA 也可用来生成一个随机位流，或加解密文本。

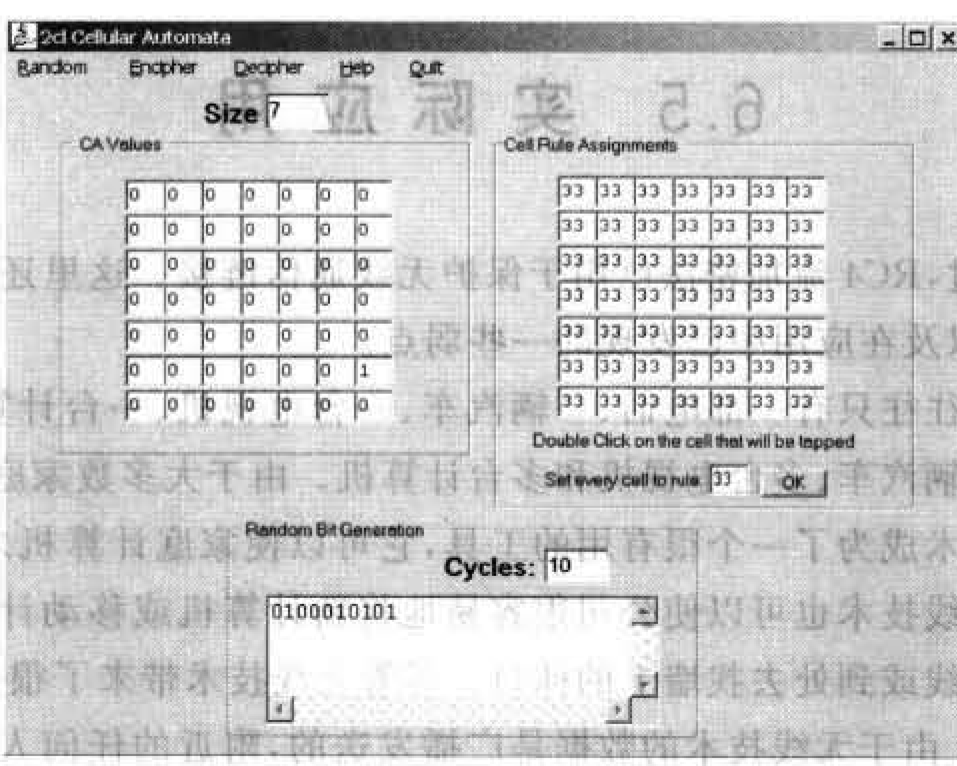


图 6.24 2d Cellular Automata 窗口

6.3.5 生成随机数的其他方法

有效随机数生成器是根据种子或密钥值，并基于一定的算法，来生成随机位流的。这其实并不是真正具有随机性的。但是，有一些随机数生成法在其中使用了一些随机事件，并利用它们来生成随机密钥流。

例如，计算机键盘上的打字方式。因此，通过监控敲击键的时间间隔，并抽取出最没意义的位，就可能生成一个小的随机位流。之所以说这个位流小，是因为我们不可能要求用户为了生成一个密钥而输入一个大文件。

还有其他一些计算机操作可用来生成随机位。鼠标移动(包括移动的时间间隔或每次移动的距离)可以用来生成随机数，这很类似于键盘输入。读取硬盘数据块所需的不同时间也是随机值，因为硬盘的磁头位置和转动速度(它会受到空气流动的影响)是变化的，这是一个非常有效的随机数生成器。

通常，随机的物理事件可以是随机位生成的来源。改革计数管记录的放射性衰变就可以看作是随机的。无线电频道上的白噪音也是随机数的来源。

6.4 一种不可破解的加密法

事实证明，在满足一定条件的情况下，流加密法是一个不可破解的加密法。如果密钥流是完全随机的，并且与明文一样长，那么它就是不可破解的。香农证明，在这种条件下，流加密法对密文攻击法是绝对安全的。

当然，要生成真正随机的位流是很困难的。而且，要求密钥流与明文一样长，这意味着不能重复。这也是很难做到的。因此，大多数流加密法只是近似地满足这些条件。因此，它们并不是不可破解的。

6.5 实际应用

前面已经介绍过，RC4 流加密法已用于保护无线通信链接。这里还是有必要简单探讨一下它的使用原理以及在应用中已发现的一些弱点。

过去，一个家庭往往只有一部电话、一辆汽车、一台电视机、一台计算机。现在，很多家庭有了多部电话、两辆汽车、多台电视机和多台计算机。由于大多数家庭并没有将计算机连接起来，所以无线技术成为了一个很有用的工具，它可以使家庭计算机之间共享文件、打印机，甚至是上网。无线技术也可以使公司很容易地将新计算机或移动计算机加入到公司网络中，而无须重新布线或到处去找墙上的插口。尽管无线技术带来了很大的便利，但无线连接也有不利的一面。由于无线技术的数据是广播发送的，附近的任何人只要有一个接收器就可以访问该信息。因此，无线连接必须加密。

由于有很多制造商设计、制造并销售无线设备，因此需要达成一种数据传输的标准格式。这种标准就是802.11，它包含了一个**无线等效保密**(wireless equivalent privacy，WEP)协议，用来加密无线数据的传输。WEP不仅仅是RC4，它是描述如何使用RC4的完整协议。对于现代的加密法来说，该协议往往与选择加密法一样重要，因为该协议不仅保护数据不被偷听，而且还防止消息被修改(保持消息的完整性)。

为了防止无线传输被他人修改，在原始的消息位中生成一个32位的校验和，称为**循环冗余校验**(cyclic redundancy check，CRC)。这个32位的校验和是有一定要求的，即消息位M加上这个32位的校验和所形成的二进制数，能被生成器数G整除。在接收端，M加上校验和也应能被G整除。如果有余数，那么该消息被修改过了，接收端不予接收。WEP的生成器数为100000100110000010001110110110111。有关保护消息完整性的方法将在第9章中详细介绍。

经增加后的消息(即M加上CRC校验和)用密钥为K的RC4加密。国际标准是40位的密钥，但在美国有一些实现(称为WEP2)使用的是104位密钥。这里就会出现一个问题。如果两个传输使用相同的密钥，就可能将它们进行XOR逻辑运算，生成明文的XOR运算结果。知道了明文的XOR运算结果，就可以进行统计攻击法。因此，WEP协议使用了一个方法，来改变不同消息所使用的密钥，即使用一个24位的初始向量(initial vector，IV)，这样实际的RC4密钥就是IV+K了。由于初始向量是惟一的，因此也将它加入到密文中，这样，传输的数据就是IV+密文了。

从加密分析员的角度来看，攻击WEP协议与攻击RC4一样可行。例如，在一个繁忙的网络中，IV很可能重复(24位对一个完全的无线协议来说并不够)，因此可以成功进行统计攻击法。另外，事实证明，CRC校验和有一个弱点，有可能有选择地修改消息中的位，并能使结果通过CRC检验。因此，现在不仅需要选择一个功能强大的加密法，而且需要设计一个安全的协议。

6.6 流加密法历史简介

第3章中介绍的自动密钥加密法显示了流加密法的一些特征。流加密法的实际应用出现在1917年，由AT&T公司的Gilbert Vernam提出的。是他首次建议在明文流中使用XOR操作的。他的这一发现成为了后来的Vernam加密法。在Vernam加密法中，明文由穿孔纸带组成，其中穿孔表示一个标记(就相当于现在的1)，空点表示一个空位(相当于现在的0)。在穿孔纸带上，字符是由5个标记和空位序列表示的，这很像现在使用的ASCII码。Vernam加密法的密钥是一个纸带，其上有随机分布的标记和空位。明文纸带与密钥纸带并行。一次读取这两条纸带上的两个位置，如果同为标记或同为空位，那么密文纸带就为空位；如果不同，那么密文纸带就为标记。也就是说，今天的流加密法就是没有纸带的Vernam加密法。

最初，Vernam使用的是一个8英尺长的纸带环来生成密钥。Vernam的同事Lyman

Morehouse，AT&T 公司的设备工程师，认为这样生成的密钥不够长，不能提供适当的安全性。他建议使用两个不同长度的密钥纸带，用一条随机纸带来加密另一条随机纸带，并将其结果作为一个二级密钥，用于实际的明文加密。如果一条随机纸带含有 50 个字符，另一条有 49 个，那么就有 2450(即 49×50)种随机字符可用作实际的加密密钥。但是，尽管进行了这样的改善，仍然满足不了 Vernam 加密法随机密钥的需要。

贝尔实验室(就是 Vernam 30 年前提出流加密法的同一地方)的数学家和工程师克劳德·香农开始了通信研究。其研究工作的结果令人惊骇。他提出了通信理论的基础，开发了加密法的度量标准，形式化了"对手通过查看密文无法获得明文的任何东西"这一概念，并证明 Vernam 加密法是不可破解的(据说他能边骑单轮脚踏车边玩耍 4 个球)。密钥的获取必须是完全随机的，并永不重复使用(因此称为一次性纸板)。香农证明，对于一个随机的不重复的密钥，不仅加密法不可破解，就是经验丰富的攻击者也无法还原明文。

有这样一个传言，早期，华盛顿与莫斯科之间的电传热线使用的是一次性纸板。大家都知道，俄罗斯的间谍使用的就是一次性纸板来交换信息的。但是，由于需要大量的密钥，今天，一次性纸板已经不再使用了。现在人们使用的是伪随机密钥生成器。

6.7 本章小结

Bob 和 Alice 终于进入了当代加密世界。他们意识到，要领先于 Eve，就必须充分利用计算机，在二进制位一级(而不是字符一级)上进行加密。他们决定使用流加密法，将二进制的密钥流与二进制的明文流进行一定的逻辑运算后生成二进制的密文流。通过一个简单的过程后，他们很快发现他们需要确保密钥流是完全随机的。大致地生成随机位的一种方法是使用线性反馈移位寄存器(LFSR)。有了这些后，利用 CAP 软件提供的工具，Bob 和 Alice 确信他们现在有了一种安全而且现代的通信方法。

与前面一样，他们的这种自信在 Eve 的努力下又失败了。Eve 发现已知密文攻击法可以破解 LFSR，没过多久她就又可以阅读 Bob 和 Alice 之间的邮件了。Bob 和 Alice 探索生成随机位流的其他方法，但发现风险都很高。对 Bob 和 Alice 来说，该是尝试其他类型的加密法的时候了。

本章介绍的流加密法分类图如图 6.25 所示。

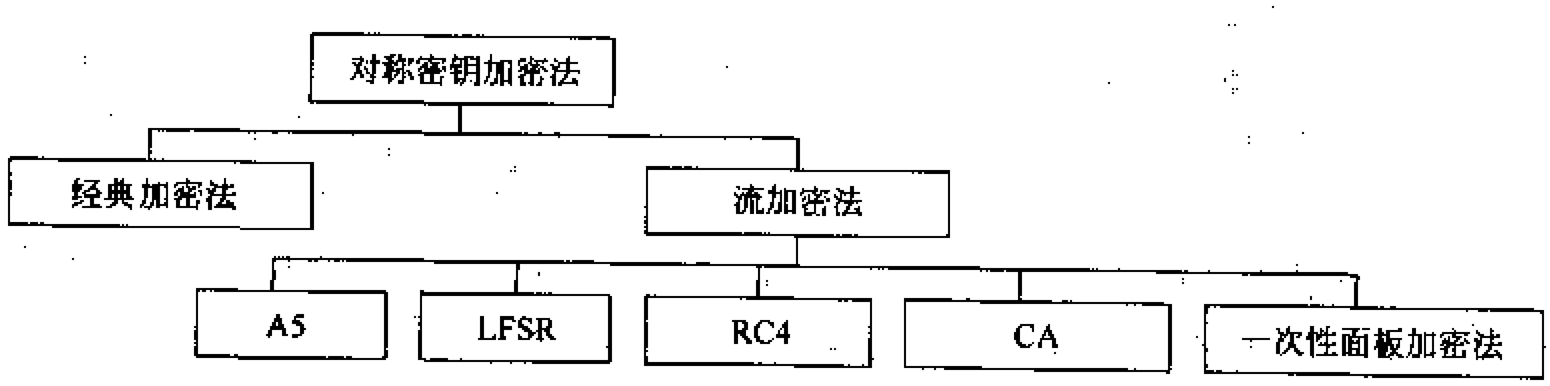


图 6.25 本章介绍的流加密法分类图

6.8 重要术语

A5	一种基于 LFSR 的流加密法，用于移动电话的安全保护。
单元自动操作	对数组中的单个单元进行操作，其中，各单元的值取决于其相邻单元和指定的规则。
CRC	循环冗余校验，一种用于确保消息完整性的方法。
LFSR	线性反馈移位寄存器，一种数组电路，它保存有 n 位。它将各位置上的位向右移动，并用选定的一些位来填充最左位。
MUX	复用器，一种数组电路，它有 2^n 个输入和一个输出，根据一定的原则选取其中一个输入作为输出。
一次性纸板	一种流加密法，其中的密钥位模式永不重复，这是惟一一种已证实是不可破解的加密法。
RC4	一种广泛使用的流加密法，它可以很容易地在软件中实现。
同步加密法	密钥流与消息流无关的加密法。
自同步流加密法	密钥流依赖于前面 n 个密文字符的加密法。
WEP	无线等效保密，该协议使用 RC4 加密法来保护无线传输。

习 题

1. 将 LFSR 全部赋为 0，看看输出流为什么。

2. 对于一个 48 位的 LFSR，有多少种可能的密钥？

3. 什么是二维 CA 中的规则 23？

4. 在一个最大长度为 4 位的 LFSR 中，以位 4 和 3 为反馈。对于从 0000～1111 的每个可能的输入，其实际长度为多少？

5. 以 $p(x)=x^5+x^3+x+1$ 表示 5 位 LFSR 的循环结构。

6. 试证明，在 A5 中，每个循环中至少有两个寄存器进行了移位。

7. 对于 $n=5$ 的 RC4 实现，有多少种可能的密钥？

8. 给定以下密文流，假设已知单词“security”出现在了明文的某个地方，而且很可能是使用 8 或 12 位长的 LFSR 来生成密钥流的：

00001010000000110101011001101100100000001

1101010001011110101101111111001111101110

1000011000100110011000110101001111001110

0010000000000110101110111001100101000101

1000011101011001101100011010011011110011

1001101000100000001110100001110111001000

00100101000001011101010111000011100000001

11001000010101011011000110100110111111111

10000000000111100011110010001011011001111

00100110010001011110111010110100010011 01

110011100101100010110001

请还原明文，并给出 LFSR 的结构。

9. 请修订 WEP，并在这个过程中大事改革无线安全性。将你的工作结果卖给出价最高的买家，并记得将一定比例的报酬给你的老师和本书作者。

第7章 块加密法

7.0 概 述

在今天所使用的加密法中，块加密法是最常见的类型。它们是从替换一换位加密法到计算机加密的概括。正如其名字所表示的那样，块加密法每次作用于固定大小的位块(而流加密法正好相反，它一次只加密一个位)，如图 7.1 所示。

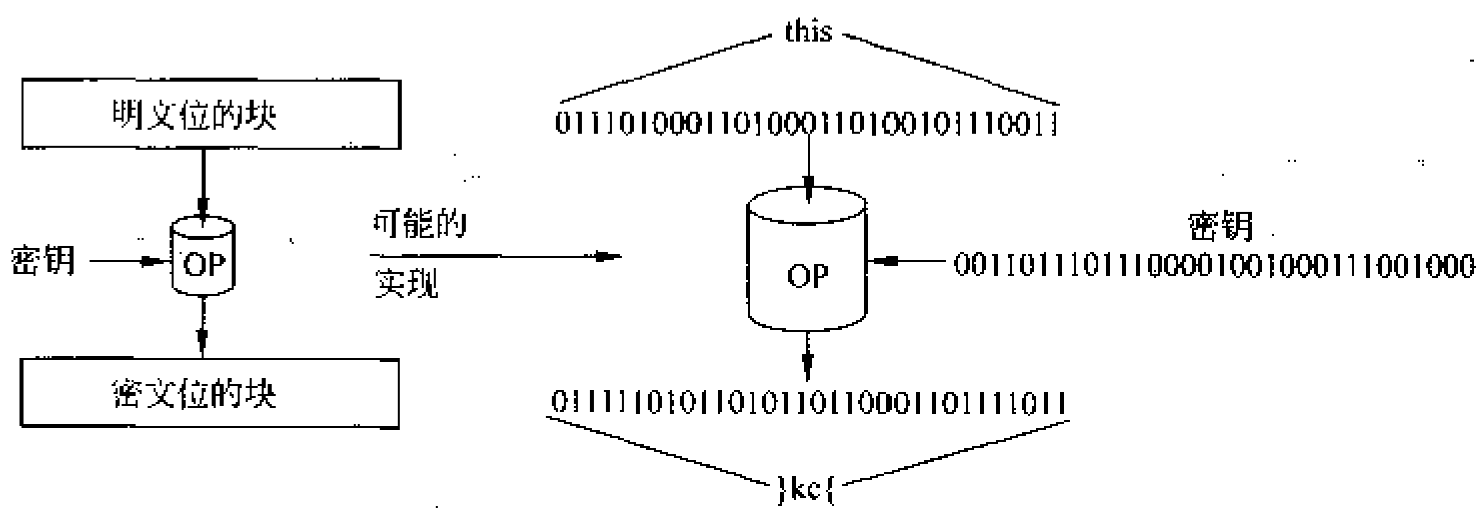


图 7.1 块加密法的常见结构

块加密法将明文分成 m 个块：$M_1, M_2, \cdots, M_m$。它对每个块执行相同的变换，从而生成 m 个密文块 $C_1, C_2, \cdots, C_m$。块的大小可以是任意数目的位，但通常是很大的数目(为 64 位或更多位的倍数)。例如，在图 7.1 中，块加密法以每块为 32 个位的方式接收明文单词"this"。它以一个 32 位的密钥在块上操作，生成 32 位的块的密文。密文表示的字符串为"}kc{"。明文的下一个 32 位块将映射到密文的另一个 32 位块。这些加密法是对整个明文操作的，包括空格、标点符号和特殊字符，而不仅仅是字符。

对于一个大的块，块加密法很像经典多图加密法，如 Playfair 加密法，也是将两个或多个明文字符映射到一个密文块。因此，块加密法具有多图加密法那种抵抗频率分析攻击法的能力。另一方面，如果块的大小只有 8 位(即是一个字符)，那么块加密法就不比经典单码替换加密法好到哪里去了。

看上去一些经典加密法(如 Vigenere 加密法)也可以称为块加密法，因为 Vigenere 加密法的关键词也可以定义块的大小。例如，对于关键词"next"，下面的 Vigenere 加密就像是对块大小为 4 的块加密：

next next next next next
this isno tago odci pher
glfl vwkh gcdh bhzb clbk

这些加密法不是块加密法的原因是，在块加密法中，密文块的所有位与明文块的所有位有关。正是这个原因体现了现代块加密法的最重要特征。如果明文的单个位发生了改变，那么

密文块的位平均有一半要发生改变。这种情况在 Vigenere 加密法中不会发生(单个位和单个字符的明显不同除外),在 Vigenere 加密法中,明文块中单个字符的改变只是导致密文块中单个字符的改变。例如,如果第一个明文块从"this"改变成了"thes",密文块将从"glfl"变成了"glbl"。

7.1　块加密法的模式

在介绍具体的块加密法之前,有必要看看块加密法的常见使用方法。这样,应用方法就没有与特定的实现捆在一起,任何块加密法(这样的加密法很多)可以以三种标准模式之一使用:电子编码簿模式、密文块链模式或输出反馈模式。其实并不止这三种模式,但这是三种讨论最普遍的模式。事实上,美国国家标准与技术协会推荐了 5 种模式。在下一节中将会介绍,一些新的模式正在吸引人们更多的注意。

电子编码簿模式(electronic-codebook mode,ECB)是最简单的模式。事实上,本章开始介绍的就是这种模式,明文的每个块加密成密文的一个块。图 7.2 演示了这种操作模式。

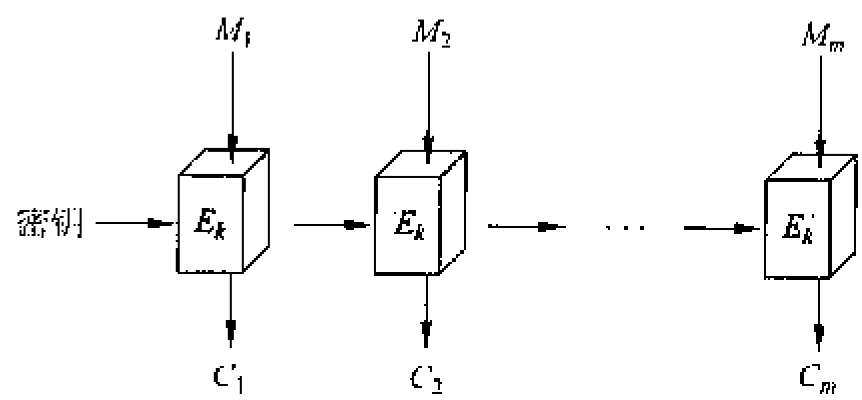


图 7.2　电子编码簿模式

加密-块-链(cipher-block-chaining,CBC)模式的实现更复杂(这是为了增加安全性)。由于更安全,因此它是使用最普遍的块加密模式。在这种模式中,来自上一块的密文与当前明文块做 XOR 逻辑运算,其结果就是加密的位块。这有点像第 3 章所介绍的自动密钥加密法。图 7.3 演示了这种模式的操作。在该图中,第一个明文块与 0 向量做 XOR 逻辑运算。这是 CBC 加密的最早方法,但不安全。CBC 更安全的实现是使用**初始向量**(initial vector,IV)。

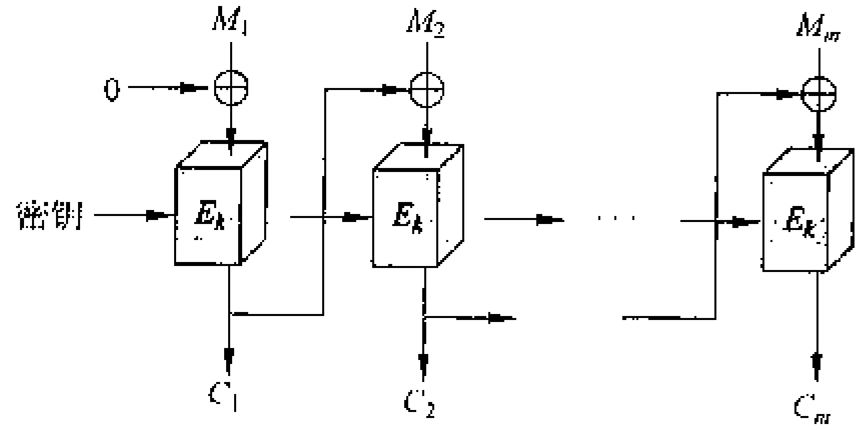


图 7.3　加密-块-链模式

如果为每个消息传输选取不同的 IV,那么两个相同的消息即使使用相同的密钥,也将有不同的密文。这大大提供了安全性。但问题是:接收端如何知道所使用的 IV 呢?一种方法是在一个不安全的通道上来发送该 IV。在这种情况下,IV 只使用一次,且永不重复。

另一种更安全的方法是基于**惟一数**(nonce,*n*umber used *once*)的概念。惟一数是一个惟一的数字,永不重复使用相同的密钥。它不一定非得保密,它可以是消息的数目等。用块加密法将惟一数加密后生成 IV。如果惟一数附加到了密文的前面,接收端就可以还原 IV。

输出-反馈模式(output-feedback mode,OFM)使用块加密法来为流加密法生成一个随机位流。密钥和块加密法的初始输入启动这个加密过程,如图 7.4 所示,通过将块加密法的输出加密,为流加密法提供了附加的密钥位。

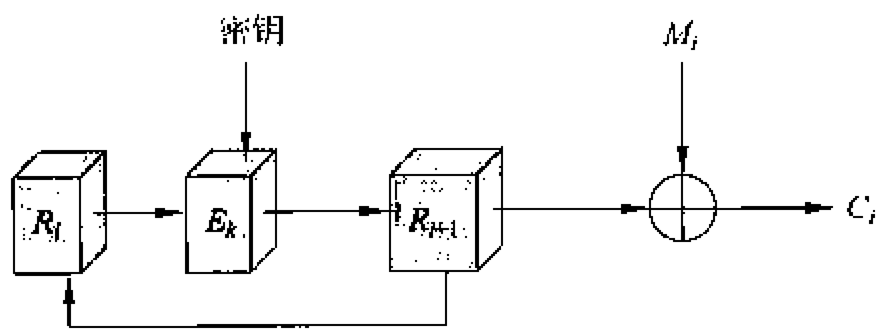


图 7.4 输出-反馈模式

7.1.1 现代块加密法的模式

人们开发了多种其他的块加密模式,用来替代这三种经典的操作模式。事实上,这是一个很有趣而且收获颇丰的研究领域。开发新的块加密模式可以当前块加密算法的安全性。

一种比较新的模式是**计数器模式**(counter mode,CTR),它最近已被采纳为 NIST 标准之一了,因此正受到很多的关注。这是另一种流加密实现的模式,很像 OFM。计数器模式如图 7.5 所示。注意,在该模式中,没有使用块加密法去加密明文,而是用来加密计算器的值,然后再与消息块进行 XOR 逻辑运算,这样,它就具有了流加密法的所有特征。计算器被更新和加密后,再与第二个消息块做 XOR 逻辑运算,以此类推。这种方法的一个很好特征是,如果同时知道了 *m* 个计算器的值,那么就可以并行地将所有消息块加密(和解密)。

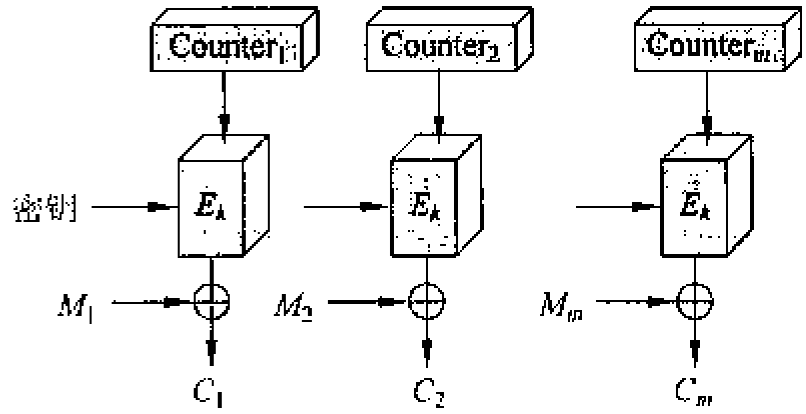


图 7.5 计数器模式

与 CBC 模式一样,CTR 模式也要求有一个初始向量(IV),用它作为第一个计算器的值(其他计算器的值可以由此 IV 值计算而来)。该 IV 值应是一个惟一数。关于 IV 选择的方法有很多种。一些加密法的 IV 值是将计数器的值与惟一值连接而成的,其他加密法的 IV 值则是从时钟、消息块数或循环计数器中获取而来的。

另一种新的模式是**补偿编码簿模式**(offset-codebook,OCB)。这种模式不仅描述了如何加密每个块,还提供了一种验证方法(有关验证方法的内容将在第 9 章介绍)。从某种程

度上来说，OCB 是目前所介绍的 5 种模式中最为复杂的一种。其操作说明如图 7.6(a)和图 7.6(b)所示。

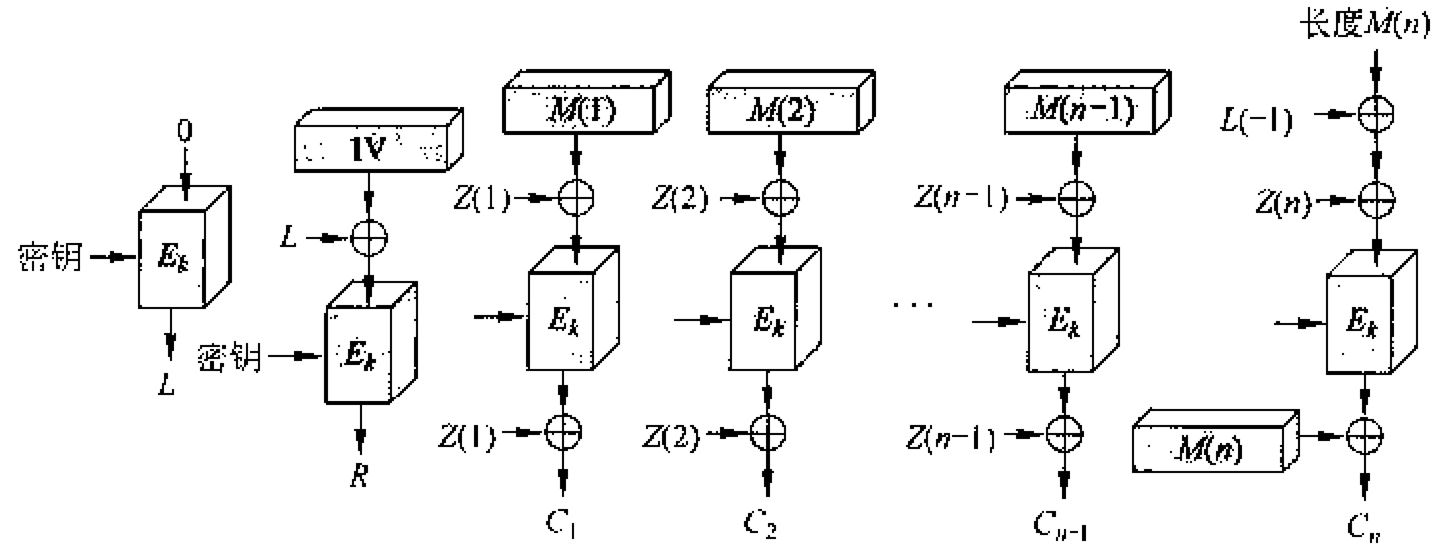


图 7.6(a) OCB 模式(部分 1)

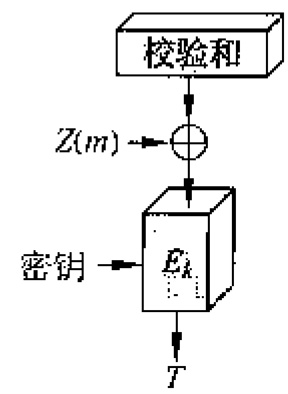


图 7.6(b) OCB 模式(部分 2)

假设一个消息有 n 个块，即 $M(1) \sim M(n)$。该模式首先用密钥将零向量加密，生成 L。L 与 IV(惟一数)进行 XOR 逻辑运算，其结果加密后生成 R。再用 R 与 L 生成二进制向量 Z，其中：

$$Z(1) = L \text{ XOR } R$$

$$Z(i) = Z(i-1) \text{ XOR } L[\text{ntz}(i)]$$

函数 $\text{ntz}(i)$ 为在 i 的二进制表示中 0 的个数。$L(1) \sim L(n)$ 的值是通过有条件地移动 L，并根据块的大小与这两个值中的一个进行 XOR 逻辑运算而生成的。

为了验证(也就是说，为了保证密文不被修改)，OCB 模式在传送密文中加入一个标志值。这个标志值是图 7.6(b)生成的 T 向量中的前 t 个位。用来生成 T 的校验和是第 m 个密文块(C_m)与 $Y(m)$ 进行 XOR 逻辑运算的结果。其中，$Y(m)$ 是第 m 个块的长度与 $L(-1)$ 进行 XOR 逻辑运算的结果。通过将所接收的校验位与重新计算的这些位进行比较，就可以实现验证。如果两者匹配，那么明文就通过验证；如果不，那么明文就不可靠。

尽管这里只是简要地介绍了一下 OCB，还没有深入到加密法的本身中去，但仍可以看得出，它是一个很复杂的处理过程。OCB 已经应用到加密法的实现中了。从这里可以看出，尽管现代加密法是扎根于经典加密法的，但它们在保证安全性上已经取得了长足进步。

7.1.2 关于填充问题

所有块加密法，不管本质如何，都必须处理一个问题——填充。出现这个问题是因为，块加密法是作用于有固定大小的块的。但如果明文的大小不是块大小的整数倍的话，该怎

么办？例如，块的大小可能是 64 位(即 8 个字符)，但明文的大小只有 96 位(即 12 个字符)。在这种情况下，第二个块就只有 32 个明文位。对大多数经典加密法来说，这不是一个问题。在 Vigenere 加密法中，如果明文长度不是关键词长度的整数倍，那么最后一次就只使用关键词的一部分。在 Playfair 加密法中，如果明文字符为奇数，就需要填充，它是通过填充惰性字母(如“x”或“q”)来解决这个问题的。在换位加密法中，填充问题就更常见。美国内战时期，北方军使用无意义的单词来填充。列换位加密法要求使用惰性字符来填满列数组。经典加密法不用太担心填充问题。但是，对自动的块加密法来说，填充就是一个问题了。在块加密法中，如果填充可逆则可行。也就是说，必须在加密时能添加填充字符，而在解密时能检测出填充字符。常见的解决办法是为明文添加(填充)足够的 0，从而使明文长度是块大小的整数倍。这里的问题是，它可能不是可逆的。例如，对于明文字母“p”与添加了一些“0”后的明文字母“p”，经加密后，再解密都是得到“p”。这就无从知道明文是否带有“0”。

Ferguson 和 Schneier 提出了两种可逆的填充方法。一种方法是用 128 位的字节后跟足够的 0 来填满块。另一种是利用所需的填充字节数 n，给每个具有值 n 的块填充 n 个字节。

7.2 乘积加密法

如果只是块加密法或以前面介绍的 5 种模式实现的，那么加密法就不弱但也不强。属于块加密法之类的经典加密法具有很多弱点。现代的块加密法也有功能强和功能弱的。块加密法的实现方法最终决定了它的性能。现今很多功能强大的块加密法都是以乘积加密法来实现的。这些加密法包含了多种加密方法。

1973 年，Feistel 提出了一种乘积加密法，它成为了今天所用的对称块加密法的体系结构基础。整个处理过程包括多个阶段的替换以及换位。主密钥分成一个子密钥集，每个阶段使用一个子密钥。在每个阶段中，数据块分成了左右两部分，这些部分进行交换，其中一个部分与子密钥混合，用于该阶段的操作。Feistel 加密法的其中一个阶段的操作如图 7.7 所示。图中的 F 块用来为加密法的每个阶段生成一个子密钥。这种类型的加密法的另一个名字是**替换-置换**(substitution-permutation)加密法。

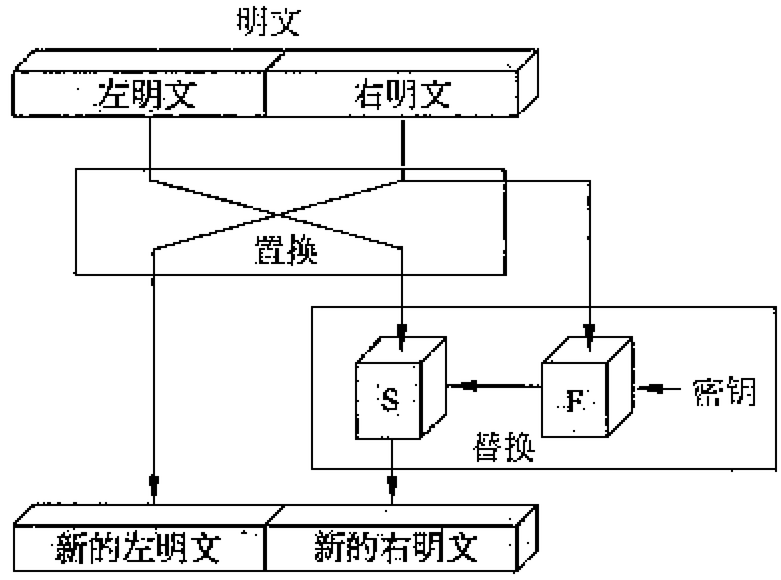


图 7.7 Feistel 加密法的第一个阶段

Feistel 加密法的可变特征有：块大小、密钥大小、轮数、轮函数以及子密钥生成过程。通过改变这些设计特征，从而在加密法的安全性和实现效率之间取得平衡。例如，密钥和块越大加密法就越安全，但加密法的实现就越慢。

7.2.1 块加密法的评估

任何新的加密法必须安全，能防止被人破解。但随着加密法变得越来越复杂(诸如 Feistel 之类的块加密法)，如何才能确信这些加密法能保护我们那些有价值的数据呢？正如 Alice 和 Bob 已经知道的那样，光凭看上去复杂的算法，并不意味着 Eve 就找不到破解它的方法。遗憾的事实是，这个问题的真正答案是我们永远无法确保一个加密法是安全的。要想知道一个新的加密法是否安全，最好办法是让安全社区的人去测试它。高级加密标准 AES 就是这样选择的，有关内容后面将介绍。

但是，如果要使一个加密法让用户可接收，那么该加密法就必须具有某些特征。首先，密钥必须足够大，使强力攻击法无效或至少是得不偿失。这就是恺撒加密法存在的问题，导致了该加密法的弃用和关键词加密法的开发。其次，如果加密法生成的密文通过了随机测试，那么在很长一段时间里，该加密法将给人以安全性。

有一些更加正式的新加密法评价标准。其中最重要的一种称为**雪崩条件**(avalanche condition)，即任何输入位或密钥位与输出位之间不应有任何连续。换句话说，密文中不能有内容含有关于密钥或明文的提示。这很重要，如果有人试用不同密钥，当尝试的密钥其中的某些位接近于实际的密钥时，不能有任何信息提示。猜测的密钥与实际密钥即使是只有一位之差，也应产生迥乎不同的密文。这里有两种版本的雪崩条件：

- 精确密文雪崩标准(strict plaintext avalanche criterion，SPAC)：无论密文块的任一位有补，密文块每个位发生改变的概率为 50%。
- 精确密钥雪崩标准(strict key avalanche criterion，SKAC)：对于一个固定长度的明文块，当密钥的任意一位发生改变时，密文块每个位发生改变的概率为 50%。

在第 4 章中，如果你完成了问题 1，那么你实际上已经为 Hill 加密法完成了 SPAC。当引入块加密法时，都必须考虑有关雪崩条件下的算法性能。

7.3 数据加密标准

20 世纪 70 年代中期，美国政府认为需要一个功能强大的标准加密系统。美国标准局提出了开发这种加密法的请求。有很多公司着手这项工作并提交了一些提议。最后 IBM 的 Lucifer 加密系统获得了胜利。1977 年，根据美国国家安全局的建议进行了一些修改后，Lucifer 就成了数据加密标准或 DES。20 多年来(比任何加密法使用的时间都长很多)，DES 都是很多应用选用的加密法。后来 DES 被**高级加密标准**(advanced encryption standard，AES)所取代。有关 AES 的详细内容将在本章后面介绍。DES 用一个 64 位的密钥来加密每个块长度为 64 位的明文，并生成每个块长度为 64 位的密文。DES 是一个包含了 16 个阶段的替换-置换加密法。

7.3.1 DES 密钥

尽管 DES 密钥有 64 位长，但用户只提供 56 位（通常是以转换成 ASCII 位的 7 个字母的单词作为密钥）。其余 8 位由算法提供，分别放在 8、16、24、32、40、48、56 和 64 位上。结果是每个 8 位的密钥包含了用户提供的 7 位和 DES 确定的 1 位。添加的位是有选择的，以便使每个 8 位的块都有奇数个奇偶校验位（即 1 的个数为奇数）。

16 个阶段中的每个阶段都使用一个 48 位的密钥，该密钥是从最初的 64 位密钥派生而来的。该密钥要穿过 PC-1 块（permuted choice 1，交换选择 1）。PC-1 块负责取出由用户提供的 56 个位。这 56 位分成左右两半。每一半都左移 1 或 2 位（视每阶段的情况不同）。新的 56 位用 PC-2（permuted choice 2，交换选择 2）压缩，抛弃 8 位后，为某个阶段生成一个 48 位的密钥。该过程如图 7.8 所示。

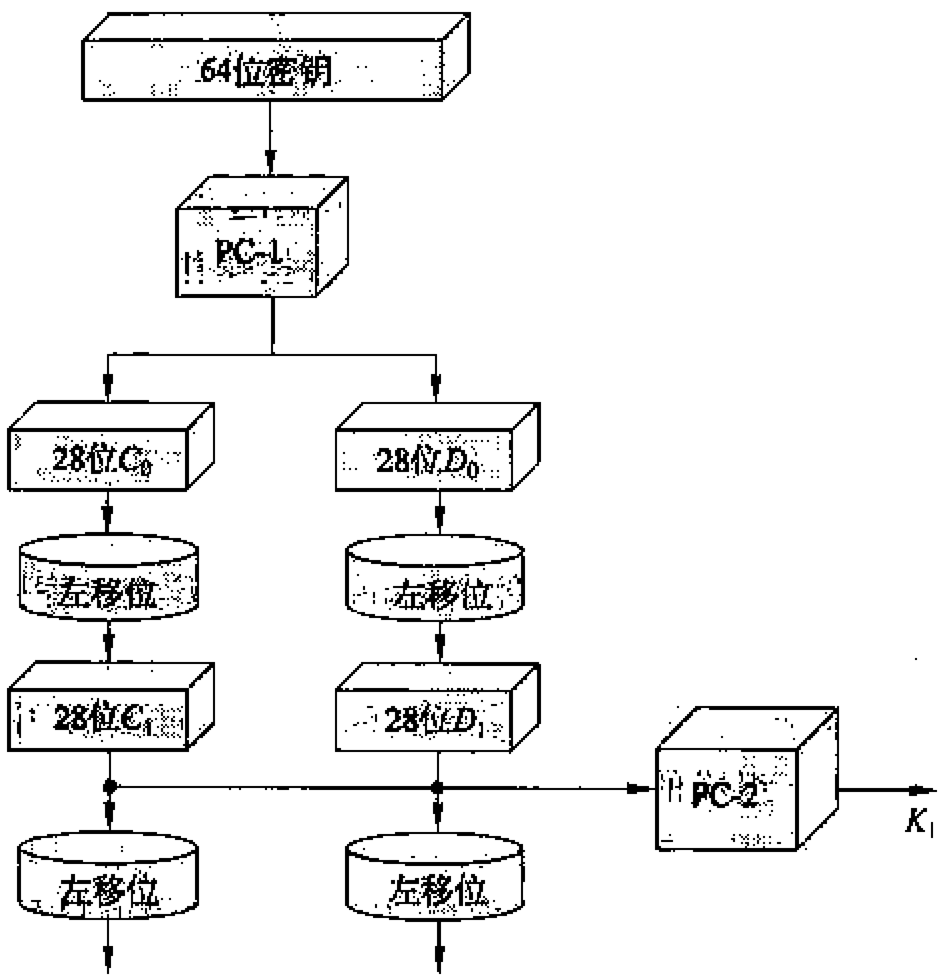


图 7.8 DES 密钥的初始阶段

PC-1 从密钥中选取 56 位，并按照如下方式重新排列：

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

PC-1 从 C_i 和 D_i 的 56 位中选取 48 位，并按照如下方式重新排列：

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

每阶段左移的位数不同，具体如下：

阶段数	左移位数
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

这是块加密法的另一个特征——密钥的操作非常精巧，这是经典加密法所不具备的。在经典加密法中，密钥就是密钥，但在块加密法中，密钥随明文的每次置换而不同。这就允许加密法的每个阶段使用不同的密钥来执行替换或置换操作。

7.3.2 DES 的各个阶段

DES 的每个阶段使用的是不同的子密钥和上一阶段的输出，但执行的操作相同。这些操作定义在三种“盒(box)”中，分别称为扩充盒(expansion box，E 盒)、替换盒(substitution box，S 盒)和置换盒(permutation box，P 盒)在 DES 的每个阶段中，这三个盒的应用顺序如图 7.9 所示。

由于每个阶段都很复杂，让我们来看看一个 64 位块通过 DES 中某个阶段的情况。我们的示例块是一个由 0 和 1 组成的。由于输入块是已知的，因此我们可以来看看该块经过 DES 的一个阶段后的变化情况。输入为 11001100 11001100 11001100 11001100 01010101 01010101 01010101 01010101。左边 32 位保留，以用于该阶段的最后一个操作中。右边 32 位作为 E 盒的输入。

通过复制一些输入位，E 盒将 32 位的输入扩充位 48 位。E 盒的扩充方式如图 7.10 所示。注意，第 1 和 32 位出现了两次，即最前面的两位和最后面的两位。

在我们的示例中，右边 32 位输入为：

01010101 01010101 01010101 01010101

这些位映射到 E 盒中就是：

1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	1	0

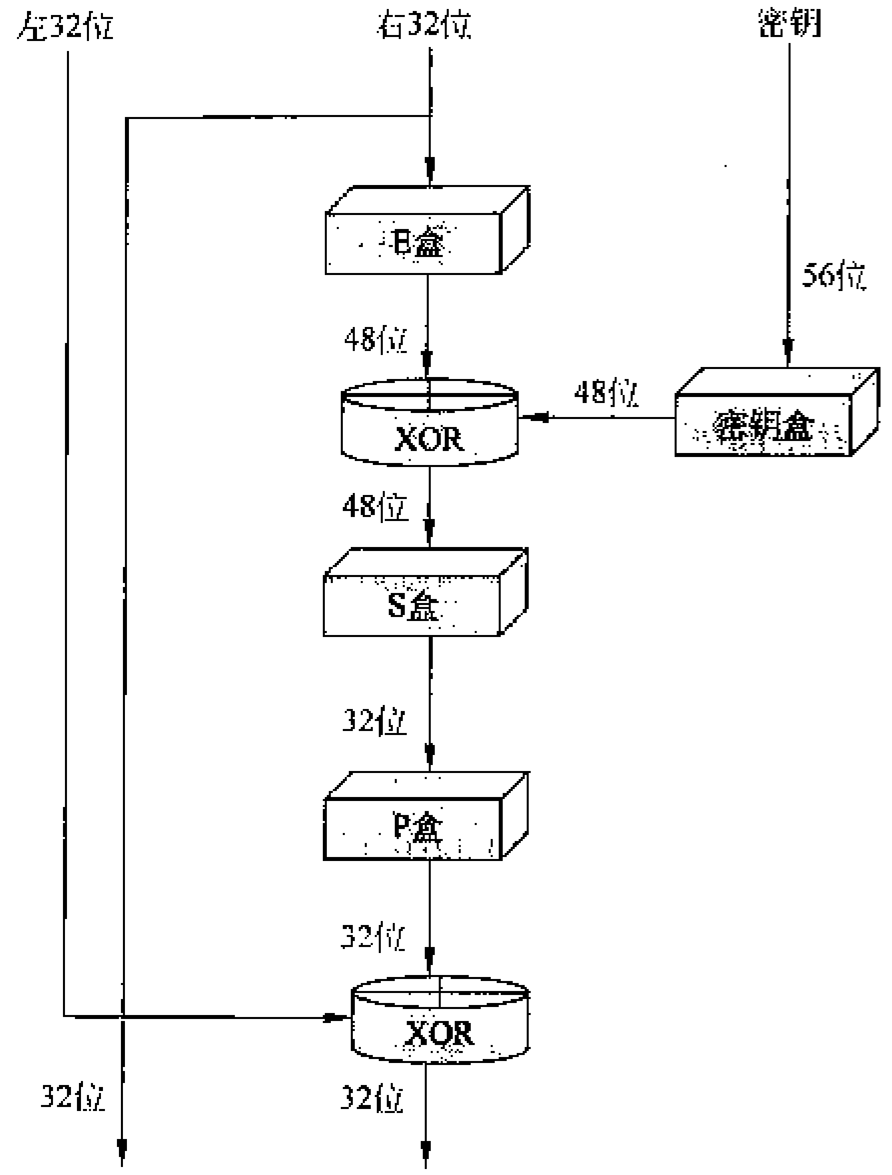


图 7.9 DES的某一个阶段

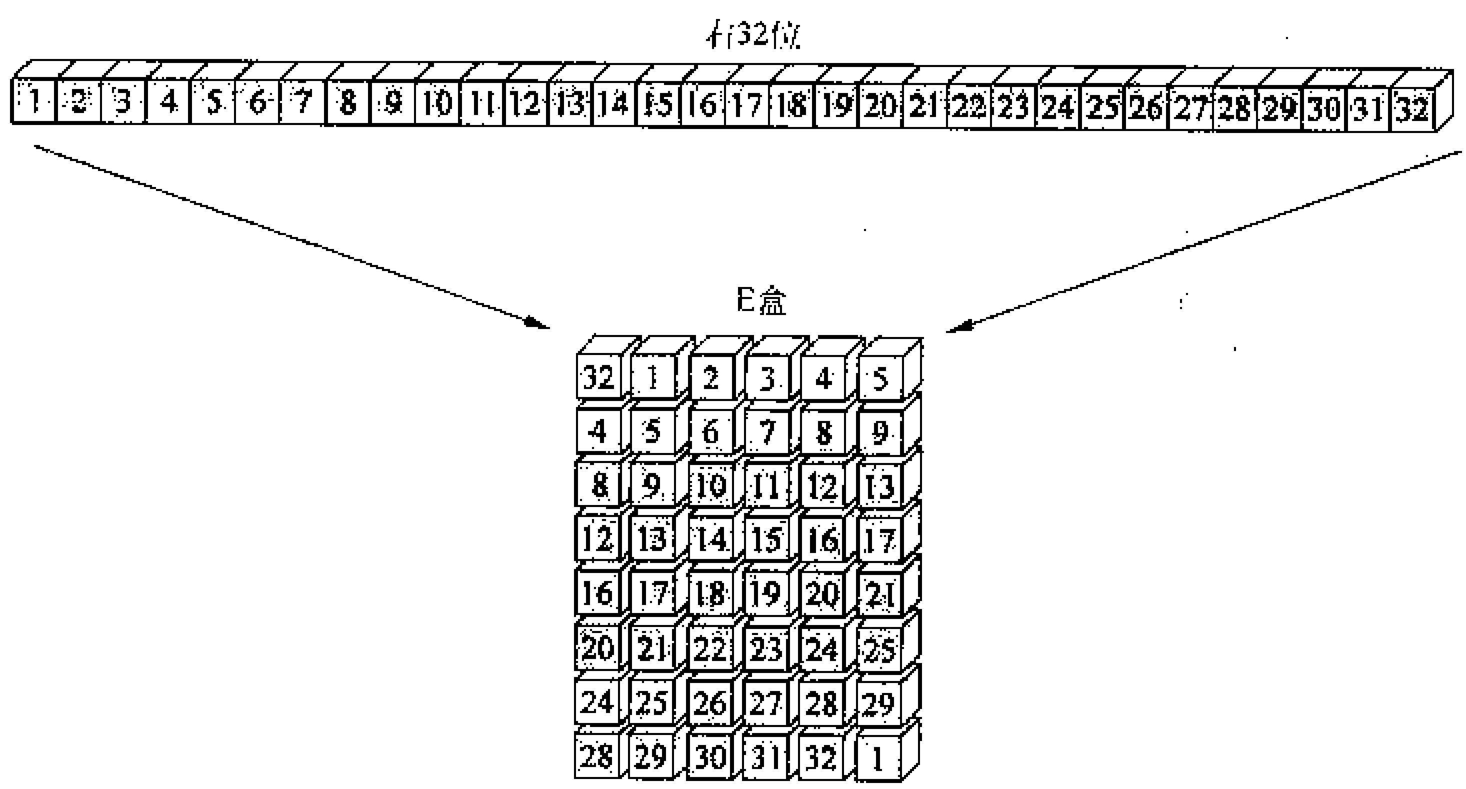


图 7.10 E盒的置换与扩充

生成如下的48位输出:

101010 101010 101010 101010 101010 101010 101010 101010

尽管这看上去数据块已经做了很大的转换,但它实际上还是很不安全的。E盒的真正作用是确保最终的密文与所有的明文位都有关。例如,后面将会看到第1位通过

E 盒操作的情况。每阶段所进行的实际转换这里不进行详细讨论，这里要考虑的是 E 盒如何在文字中“移动”初始位。第一个 E 盒操作将位复制，并将它放到位置 2 和位置 32。第二次 E 盒作用于该单词，初始位的影响延伸到了位置 1、3 和 31。等到第 8 次该单词通过 E 盒后，可以注意到 E 盒对每个位都有影响(证据是每列都有一个或多个“1”)。由于在每阶段中左右两边都进行了交换，16 个阶段后就可发生这种结果。以下是 E 盒的操作：

初始	10000000000000000000000000000000
	只有位的值有影响
第 1 阶段	010000000000000000000000000000001
第 2 阶段	10100000000000000000000000000010
第 3 阶段	01010000000000000000000000000101
第 4 阶段	101010000000000000000000000001010
第 5 阶段	010101000000000000000000000010101
第 6 阶段	10101111010000000000001010101010
第 7 阶段	01010101111000100101010101010101
第 8 阶段	10101010101111111010111010101010

下一步操作是将 E 盒的输出与 48 的子密钥进行 XOR 逻辑运算。如果子密钥是有如下的 48 位模式组成的：

输入	101010	101010	101010	101010	101010	101010	101010	101010
密钥	001100	001110	110010	001000	001100	100011	001101	110001
输出	100110	100100	011000	100010	100110	001001	100111	011011

该操作将输出一个新的 48 位块，该块作为 S 盒的输入。S 盒是 DES 强大功能的源泉。这些盒定义了 DES 的替换模式。有 8 种不同的盒。S 盒 1 如图 7.11 所示。每个 S 盒接收一个 6 位的输入，输出一个 4 位的输出。一个 S 盒有 16 列和 4 行，它的每个元素是一个 4 位的块，通常用十进制表示。

列

行	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

图 7.11 S 盒 1

例如，S 盒的第 1 行第 5 列为十进制数 2，其实际的二进制表示为 0010。注意，S 盒的列号为 0～15，而行号为 0～3。每个 6 位的输入分成一个行索引和列索引。行索引由位 1 和 6 给定，位 2～5 提供列索引。

DES 中使用的特殊 S 盒不仅仅是在其他块加密法中使用的替换。为 DES 选用这些特殊 S 盒的原因仍是保密的。但是，查看 DES 的 S 盒结构，就可以发现一些加密法的特

征。例如,S盒的每行是列号的一个非线性映射。考虑S盒1的行0,列与S盒值的关系如下:

列	值
0	14
1	4
2	13
3	1
4	2
5	15
6	11
7	8
8	3
9	10
10	6
11	12
12	5
13	9
14	0
15	7

非线性特征表明,不存在如下形式的线性方程:

$$y = ax + b$$

其中 a 和 b 为常量,x 为列号,y 为S盒的值。如果以图表的形式来表示列与S盒的值,就可以很好的说明S盒的非线性结构。DES S盒这种非线性特征意味着,给定一个输入-输出值的集合,很难预计所有S盒的输出。

DES S盒另一个很重要的特征是,改变一个输入位,至少会改变两个输出位。例如,如果S盒1的输入为010010,其输出位于行0(二进制为00)列9(二进制为1001),值为10(二进制为1010)。如果输入的一个位改变,假设改变为110010,那么输出位于行2(二进制为10)列9(二进制为1001),值为12(二进制为1100)。比较这两个值,可注意到中间的两个位发生了改变。每个输入位的改变都是如此。其影响是,输入发生了小的改变,在输出中将产生更大的改变,这可以认为是加密法的一个有用特征。

块加密法的S盒设计是一个有着广阔空间的研究领域,但是,已经有了一些基本的设计要求。例如,S盒的每行必须包含所有可能输出位的组合。如果S盒的两个输入只有一位不同,那么输出必须至少有两位不同。如果两个输入中间的两位不同,那么输出也必须至少有两位不同。

E盒的输出分成多段,每段有6位,而且每段作为8个S盒的一个输入。在我们的示例输入中,每个块的行和列值为:

S盒	1	2	3	4	5	6	7	8
输入	100110	100100	011000	100010	100110	001001	100111	011011
行	2	2	0	2	2	1	3	1
列	3	2	12	1	3	4	3	13

每个 S 盒的输出由指定的行和列给定。使用如图 7.12 所示的 8 个 S 盒，输出如下：

S1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	5
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S5

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S6

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S7

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

图 7.12 所有 8 个 S 盒

S 盒	1	2	3	4	5	6	7	8
输入	100110	100100	011000	100010	100110	001001	100111	011011
行	2	2	0	2	2	1	3	1
列	3	2	12	1	3	4	3	13
输出	1000	0111	1011	0110	1011	0111	1000	1110

现在只有 32 位，根据如图 7.13 所示的置换表，重新排列这 32 位。

输入 10000111101101101011011110001110

输出 01101101111000100111000110111101

最后的操作是将初始右半边的 32 位作为左半边，而初始左半边的 32 位与 P 盒的 32 位进行 XOR 逻辑运算，并将运算结果作为右半边的 32 位：

				11001100	**11001100**	**11001100**	**11001100**
			XOR	**01101101**	**11100010**	**01110001**	**10111101**
01010101	**01010101**	**01010101**	**01010101**	**10100001**	**00101110**	**10111101**	**01110001**

将 DES 一个阶段的输入与输出比较：

输入：11001100 11001100 11001100 11001100 01010101 01010101 01010101 01010101

输出：01010101 01010101 01010101 01010101 10100001 00101110 10111101 01110001

可以注意到，这两个位串有 33 处不同。换句话说，在单个阶段中，有约多于一半的位发生了变化。

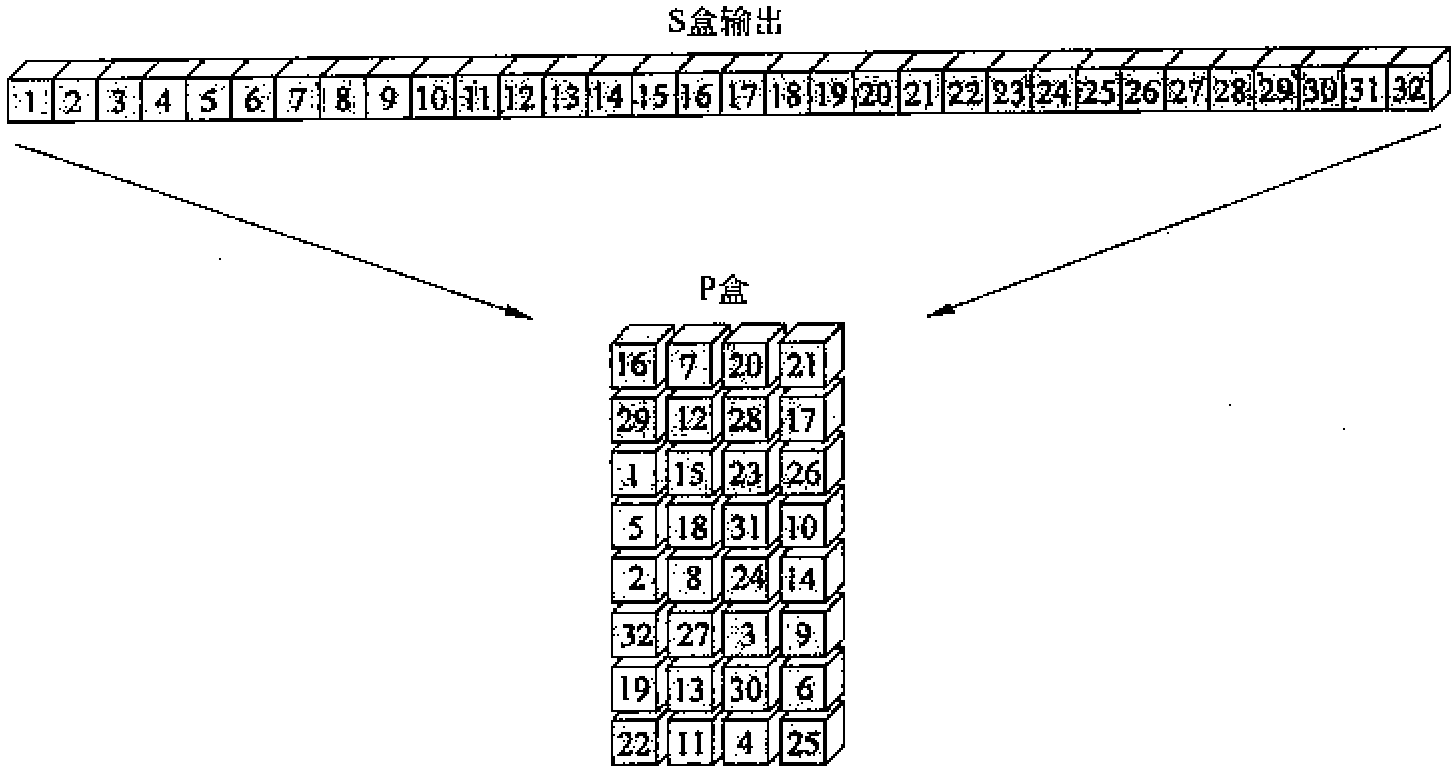


图 7.13 P盒置换

7.3.3 DES的其他实现

DES可用于5种标准块加密法的实现中：OFM、CBC、CTR、OCB或ECB。但是，正如将在7.3.5节所介绍的那样，DES不再是一个安全的加密法了。因此，提出了DES的其他实现，以提供其整体安全性。最常见的是三重DES(triple-DES)。三重DES有两个版本：一个使用了是三个密钥；另一个使用两个密钥。三密钥版本的操作过程是，先用Key1加密消息，然后用Key2解密，最后用Key3加密，如图7.14所示。两密钥版本使用的步骤与三密钥版本相同，但其中的Key3＝Key1。

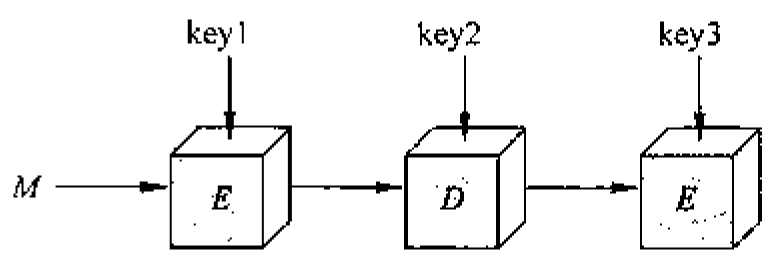


图 7.14 三重DES的三密钥版本

7.3.4 CAP软件的DES实现

CAP软件提供两个版本的DES，即一个标准的DES实现和一个小型版本(作为实验之用)。两者操作的方法相同，都是先输入明文或密文，然后选取Ciphers菜单下的DES(标准的DES实现)或S-DES(简化版的DES)。

CAP软件的DES实现还提供了一个评估选项。你可以对DES进行两种类型的雪崩条件测试。

CAP软件之所以提供S-DES(Simplified-DES)，是因为它是一个很好的学习工具。它是由Santa Clara大学的Edward Schaefer博士于1996年开发的。S-DES很简单，以便于

读者探究 DES 的操作及其一些弱点。整个结构如图 7.15 所示,其中包括了 S-DES 的密钥生成过程。它作用于 8 位的数据块(即单个字符),使用的是 10 位的密钥(因此只有 $2^{10}=1024$ 种密钥),包含有两个阶段。正如图 7.15 所示的那样,尽管简单,但它很像标准的 DES。具体操作过程是,首先进行的是将 8 个数据位 1～8 置换为 2 6 3 1 4 8 5 7。最后的逆置换是 4 1 3 5 7 2 8 6。函数 F 包含有一个 E 盒、一个 P 盒和两个 S 盒,很像 DES。E 盒按 4 1 2 3 2 3 4 1 的顺序将 4 位的输入扩充为 8 位的输出。P 盒通过置换选择生成 8 位的子密钥,该子密钥与前面 E 盒的 8 位输出进行 XOR 逻辑运算。运算结果的前 4 个位作为 S 盒 0 的输入,其他 4 个位作为 S 盒 1 的输入。这两个 S 盒如下所示:

S盒0

行＼列	0	1	2	3
0	1	0	2	3
1	3	1	0	2
2	2	0	3	1
3	1	3	2	0

S盒1

行＼列	0	1	2	3
0	0	3	1	2
1	3	2	0	1
2	1	0	3	2
3	2	1	3	0

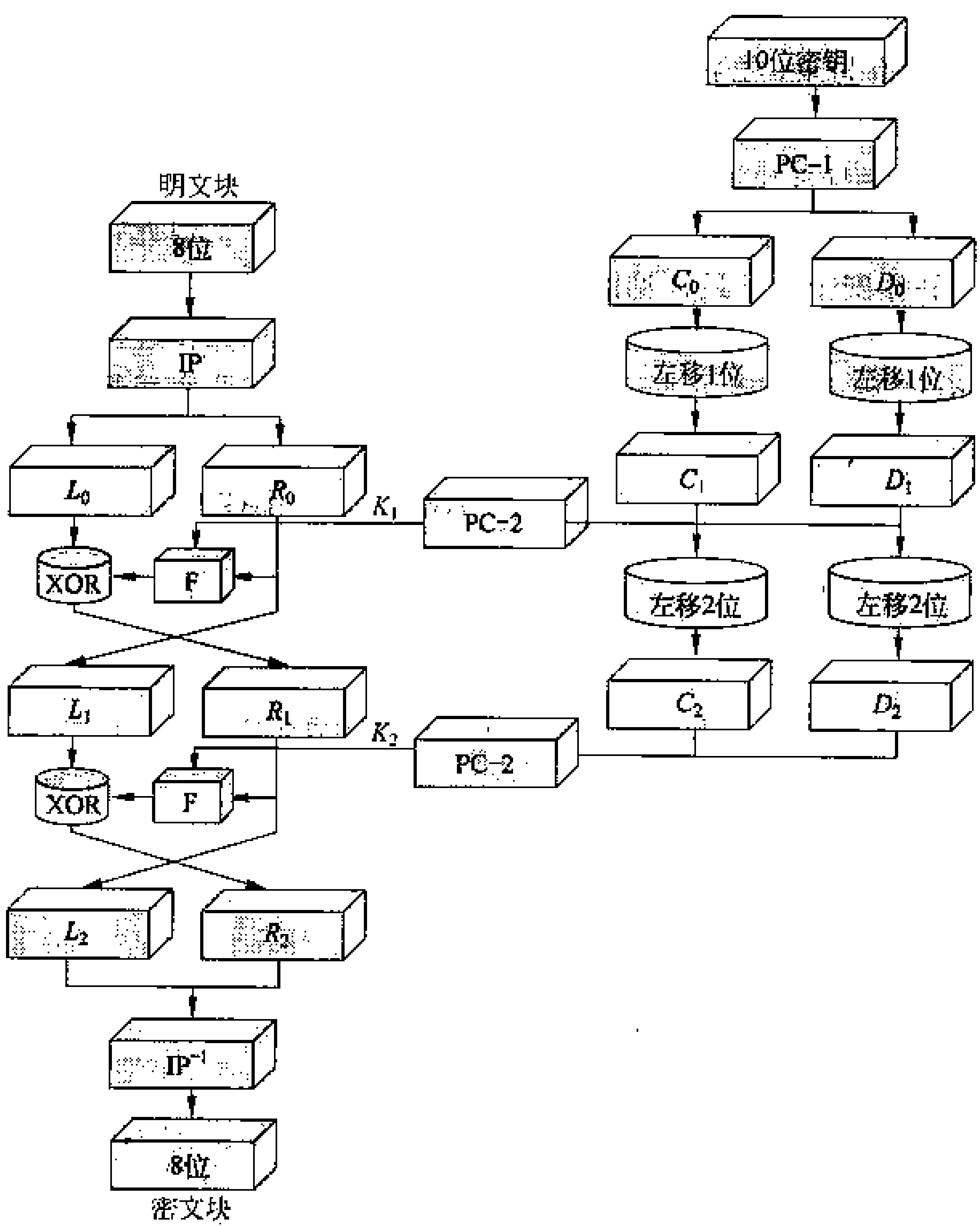


图 7.15 S-DES

每个S盒输入的第一位和最后一位确定第几行，中间两位确定第几列。每个S盒的输出是一个两位的二进制数。P盒按2 4 3 1的顺序将这两个输出进行置换。这个输出作为S-DES每个阶段输出的右半边，而初始的右半边作为S-DES每个阶段输出的左半边。最后一个阶段的输出置换后就成了密文块。

生成密钥的机制是，PC-1按3 5 2 7 4 10 1 9 8 6的顺序将10位的密钥置换。接着将置换结果分成两个段，每段有5位。将每段左移一位后，PC-2从这两个段中选取8位，并按6 3 7 4 8 5 10 9的顺序重新排列，该结果就是子密钥1。将每段左移两位后，PC-2重复应用一次，生成子密钥2。

CAP软件实现了这种版本的S-DES。选取Ciphers菜单下的S-DES项，弹出如图7.16所示的窗口。所输入的密钥必须是0～1023的整数。当输入密钥值并单击Set Key后，将显示密钥生成过程的结果。

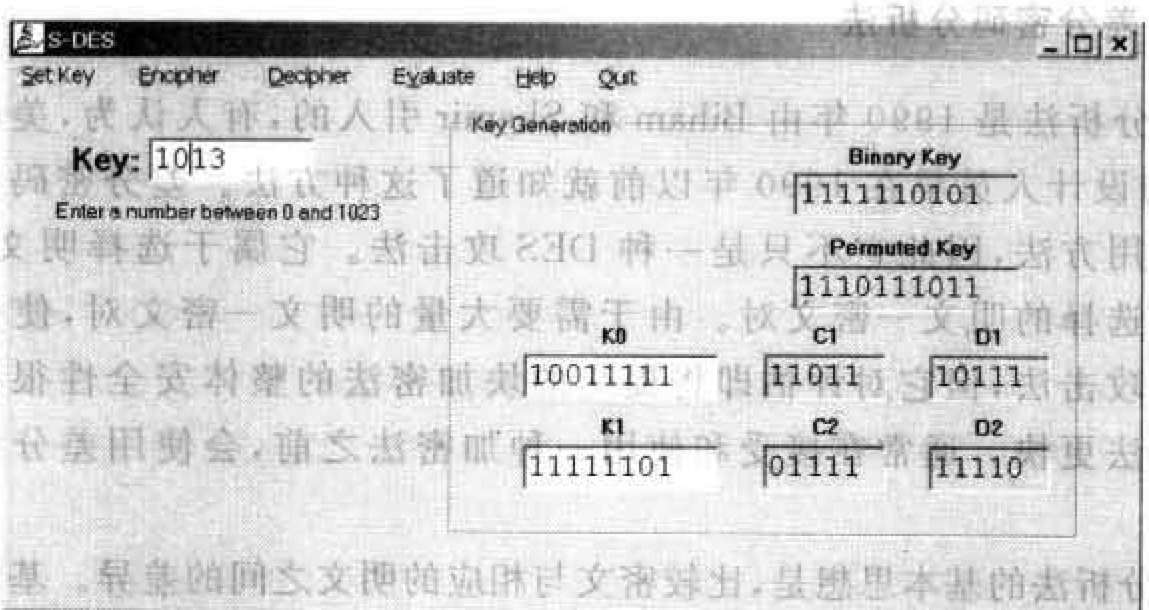


图7.16 S-DES窗口

7.3.5 DES分析

DES有着一个充满了思考和争论的漫长而有趣的过程。其起始点是美国国家安全局(NSA)要求对IBM提交的Lucifer加密法初始说明进行修改。其中之一的修改是，要求将初始密钥长度从128位缩减为56位。这使人推测(但从未得到证实)认为，NSA可以破解56位的DES。也许他们能够破解，但没有人说过，但NSA以外的其他人尝试了20多年，也取得了不同程度的成功。强力攻击法看上去是破解其算法的惟一可行方法。这也是直到20世纪90年代末期计算机技术的发展才得以实现。1997年7月，借助因特网上14000多台计算机，花费了90天所时间破解了DES密钥。6个月后，用这种方法破解DES所花费的时间减少到了39天。1998年7月，一台特殊构造的计算机(名为Deep Crack)只用了56个小时就破解了一个DES密钥。显然，DES不再是一个可靠的加密系统了，因此，美国国家标准局(NBS)提出了一项取代DES的投标计划，这就是高级加密标准(advanced encryption standard，AES)。有关AES的内容将在本章稍后介绍。

对DES结构的研究不仅导致了更快速的、用于破解块加密法的强力攻击法的开发，还刺激了新的密码分析工具的开发，如线性分析法和差分分析法。这两种方法也将在本

章稍后介绍。另外,人们还发现了DES的一些有趣特性,这些特性暗示了DES算法的某些弱点。现在,密码学的研究人员将从DES所学到的知识应用到了对新的块加密法的评估上。

较早的研究发现,DES有一些脆弱的密钥。这些密钥为每轮生成相同的子密钥。下面4个DES密钥就是这种类型的:

0101	0101	0101	0101	FEFE	FEFE	FEFE	FEFE
1F1F	1F1F	0E0E	0E0E	E0E0	E0E0	F1F1	F1F1

还有12个亚脆弱(semiweak)的DES密钥。亚脆弱的密钥只交替地生成两种密钥。当然,使用DES时,必须避免使用脆弱的或亚脆弱的密钥。但是,由于脆弱的或亚脆弱的密钥都只是少数,对DES算法的整体安全不会构成大的影响。S-DES也有脆弱的密钥。例如,密钥0就只生成相同的子密钥(K0和K1都是00000000)。

7.3.5.1　差分密码分析法

差分密码分析法是1990年由Biham和Shamir引入的,有人认为,美国国家安全局和DES的最初设计人员早在1990年以前就知道了这种方法。差分密码分析法是分析块加密法的常用方法,因此它不只是一种DES攻击法。它属于选择明文攻击法,因为它需要大量有选择的明文—密文对。由于需要大量的明文—密文对,使得该方法不是一种很实用的攻击法,但它对评估即将提交的块加密法的整体安全性很有用。它的速度比强力攻击法更快。通常在接受和使用一种加密法之前,会使用差分密码分析法来评估其弱点。

差分密码分析法的基本思想是,比较密文与相应的明文之间的差异。基于这些差异,就可以对密钥的位进行合理的猜测。下面来考察一个简单的流加密法,其中的密文是明文与密钥进行XOR逻辑运算的结果:

$$C_1 = P_1 \oplus K$$

例如:

密钥	0	0	1	1	0	1	1	1	0	1
明文	0	1	1	0	0	1	0	1	0	0
密文	0	1	0	1	0	0	1	0	0	1

如果用同一个密钥将两个不同的明文流加密,然后将得到的两个密文流进行XOR逻辑运算,结果是:

$$C_1 \oplus C_2 = P_1 \oplus K \oplus P_2 \oplus K$$

于是可以得出:

$$C_1 \oplus C_2 = P_1 \oplus P_2$$

例如,我们可能有:

K	0	0	1	1	0	1	1	1	0	1	K	0	0	1	1	0	1	1	1	0	1
P_1	0	1	1	0	0	1	0	1	0	0	P_2	0	1	1	0	1	0	0	0	1	1
C_1	0	1	0	1	0	0	1	0	0	1	C_2	0	1	0	1	1	1	1	1	1	0

这里，由于使用的是相同的密钥流，下面来看看两个密文流的 XOR 逻辑运算结果，以及两个明文流的 XOR 逻辑运算结果：

C_1 XOR C_2	0	0	0	0	1	1	0	1	1	1
P_1 XOR P_2	0	0	0	0	1	1	0	1	1	1

正如所预料的那样，两者产生相同的结果。这里的问题是，如果知道了两个密文流和其中一个明文流，那么就可以确定另一个明文流。例如，如果已知了 P_1、C_1 和 C_2，那么：

$$P_2 = C_1 \text{ XOR } C_2 \text{ XOR } P_1$$

即：

C_1 XOR C_2	0	0	0	0	1	1	0	1	1	1
P_1	0	1	1	0	0	1	0	1	0	0
P_2	0	1	1	0	1	0	0	0	1	1

由于流加密法具有的线性特性，因此明文之间的差异与密文之间的差异是相同的。同样，这两者与实际的密钥也无关。这使得可以用已知明文—密文对来破解，但这不能直接找出密钥(可以间接地找出密钥，但这不是这里要讨论的问题。)。但是，块加密法像 DES 一样不是线性的(还记得 DES 的 S 盒的非线性特性吗?)。因此，给定一个密钥，两个明文对之间的差异与它们的密文之间的差异是不同的。实际上这就是密钥所起的作用。这看上去像是一个优点，因此它去除了使用前面介绍的方法来破解的可能性，但实际上产生了另一个可能的弱点。这就是，某些差异可能会暗示密钥的值。你可能堵住了一个漏洞，但却导致另一个漏洞。这也是你在研究新的加密法时，必须深思熟虑的原因所在。

块加密法的非线性特性是由 S 盒的结构所导致的，因此差分分析法就从考察 S 盒的特性着手。例如，让我们来看看 S-DES 中的 S_0 盒。该盒的二进制输入和输出可写成如下的数组(记住，第一和最后一个输入位指定行，中间两位指定列)：

二进制输入	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
	1010	1011	1100	1101	1110	1111				
二进制输出	01	11	00	01	10	00	01	11	10	01
	00	11	11	01	01	00				

假设有两个不同的 X_1 和 X_2 输入 S_0 盒，那么 $\Delta X = X_1$ XOR X_2。如果 Y_1 为 X_1 的输出，Y_2 为 X_2 的输出，那么 $\Delta Y = Y_1$ XOR Y_2。我们需要做的是找出所有可能的(ΔX, ΔY)对，并将每个对出现的次数填入**差分分布表**(difference distribution table，DDT)中。S_0 和 S_1 盒的 DDT 如图 7.17 所示，其中行代表 ΔX 的值，列代表 ΔY 的值(以十六进制表示)。该表显示，对于 S_0 盒，不会出现这样的对：输入差分为 3，而输出差分为 0。但是，有两对的输入差分为 7 而输出差分为 3。该表还可看作是概率值。例如，如果 S_0 盒的输入差分为 2，那么输出差分为 2 的概率是 8/16 或 1/2。对于一个"理想"的 DDT，每个单元都应有相同的值(这意味着在如图 7.17 所示的表中，所有的项都应为 4)。由于 S_0 盒的 DDT 不是理想的，于是就可以用它来探究密钥。

S₀（行为 ΔX，列为 ΔY）

ΔX \ ΔY	0	1	2	3
0	16	0	0	0
1	0	4	8	4
2	0	4	8	4
3	4	8	0	4
4	2	6	2	6
5	4	4	4	4
6	4	4	4	4
7	6	2	6	2
8	4	4	4	4
9	2	6	2	6
A	6	2	6	2
B	4	4	4	4
C	4	8	4	0
D	8	0	4	4
E	4	0	4	8
F	4	8	4	0

S₁（行为 ΔX，列为 ΔY）

ΔX \ ΔY	0	1	2	3
0	16	0	0	0
1	2	8	2	4
2	0	6	4	6
3	4	2	8	2
4	2	0	10	4
5	2	4	2	8
6	0	10	0	6
7	8	2	4	2
8	4	6	0	6
9	8	2	4	2
A	2	0	10	4
B	0	6	4	6
C	0	6	4	6
D	6	0	6	4
E	10	4	2	0
F	2	8	2	4

图 7.17　S-DES 的差分分布

知道了块加密法中所有 S 盒的 DDT 只是起步而已。差分密码分析的最终目标是找出整个加密法的差分特性，即明文对差分与结果密文差分。这里以 S-DES 为例来讲解，因为它相对简单些，而且我们已经知道了两个 S 盒的 DDT 了。

图 7.18 显示了 S-DES 的第一个阶段，其中的随机输入对是有选择性的，以便 S_0 盒的 ΔX 为 1(二进制为 0001)，S_1 盒的 ΔX 为 0。从 S_0 盒的 DDT 中可知，$\Delta Y=2$ 的概率为 8/16 或 1/2。当然，对于 S_1 盒，$\Delta Y=0$ 的概率为 16/16，即 1。

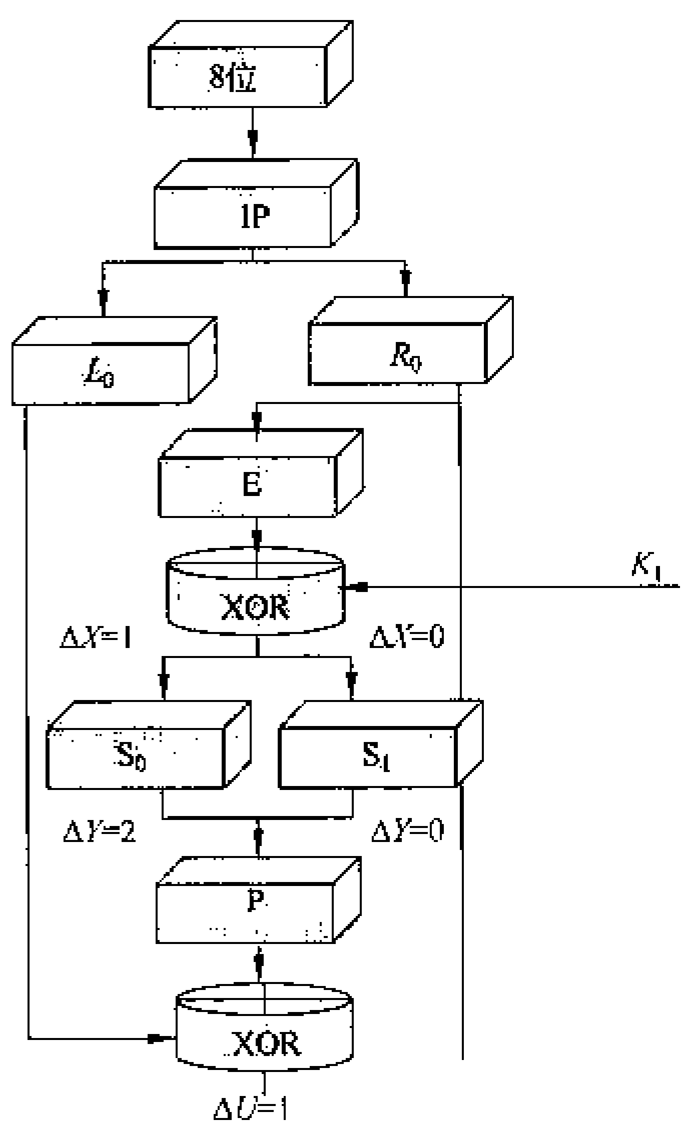


图 7.18　S-DES 的第一个阶段

于是该阶段的差分输出为1000，经P置换为：

(1000) → (0001)

第一个阶段的差分输出为1，该值输入到S-DES的下一个阶段中，如图7.19所示。第二个阶段的差分输入经E置换后分成两半，于是两个S盒的输入为4和1。对于S_0盒，输入差分为4而输出差分为1的概率为6/16(从图7.17的DDT中查找而来)。对于S_1盒，输入为1而输出为1的概率为8/16。由于有最后的置换P，因此该阶段的输出为C。

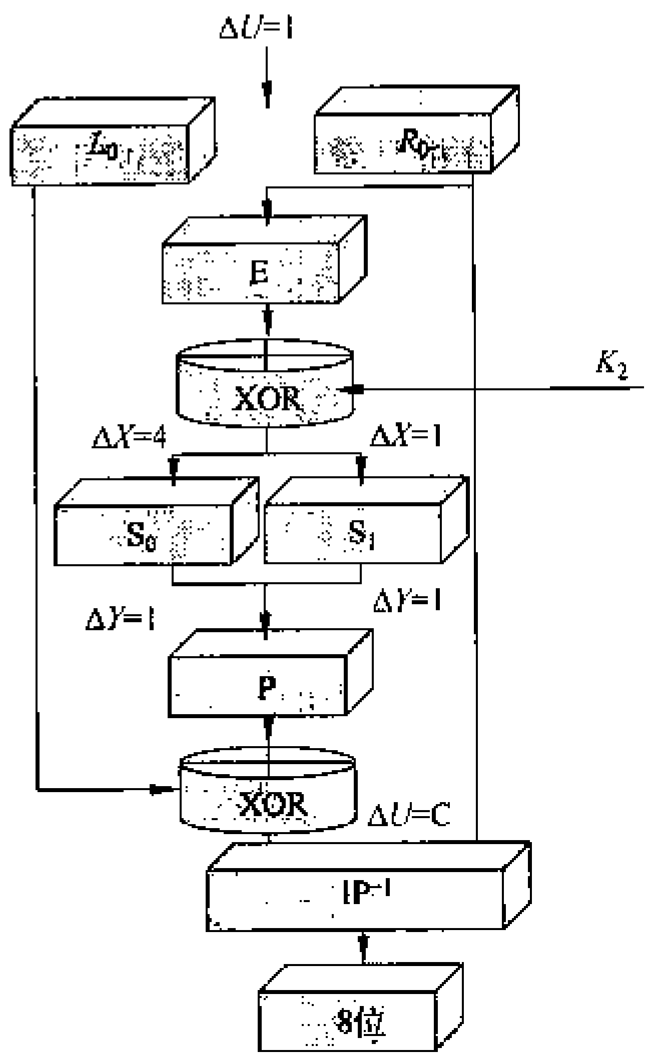


图7.19 S-DES的第二个阶段

现在，差分对$\Delta X_{in}=1$和$\Delta Y_{out}=C$就是我们的目标。它发生的概率为：

$$\frac{8}{16}\times\frac{6}{16}\times\frac{8}{16}=\frac{384}{4096}=\frac{3}{32}$$

要找出真正的密钥，需要尝试大量差分为1的随机明文对。那些具有正确输出差分(这里是指C)1，就称为**正确对**(right pair)，而那些不能产生正确输出差分称为**错误对**(wrong pair)。这些正确对可在加密法的最后一个阶段中用来猜测可能的密钥位。这可以将密文回溯到该阶段，尝试所有可能的子密钥来完成。记录下每个密钥输出正确差分的次数。尝试完所有子密钥后，次数最大的子密钥很可能就是加密法所使用的子密钥。重复该过程，知道有了足够多的子密钥信息，可以找出真正的子密钥。

由于这种方法要求穷举子密钥，这看上去比穷举主密钥好不到哪里去。其实并非如此，因为子密钥更小，因此这是一种可行的方法。例如，DES的主密钥是56位的，要求搜索2^{56}个密钥，而子密钥只有48位，因此只需搜索2^{48}个密钥。当然，差分密码分析需求大量的数据，包括大量的已知明文—密文对。

7.3.5.2 线性密码分析

线性密码分析是一种已知明文攻击法，由 Mitsuru Matsui 于 1993 年提出的。由于块加密法的非线性结构是这种加密法安全性的主要源泉，因此线性密码分析试图发现这些结构中的一些弱点。这可以通过查找非线性的线性近似来实现。这里假设用于近似算法的密钥值可用于实际算法。至少其结果能为查找实际密钥位提供有力的线索。

Feistel 加密法一个典型的阶段如图 7.7 所示。该阶段的线性部分是由密钥和明文的函数 $f(x,k)$ 定义的替换操作。线性分析是看该函数是否可写成如下的线性组合：

$$y = f(x,k) = (\mathrm{A}x + \mathrm{B}k) \bmod 2$$

其中 A 和 B 为常数矩阵。如果存在这样的 A 和 B，那么又由于知道了 y 和 x，所以就可以计算出 k 的值。

现在来看看 S-DES 的情况。如果忽略初始和最后的置换操作，那么对明文的操作就可以表示成如图 7.20 所示的函数集。函数 F 表示 S 盒的操作，函数 G 是交换函数，因此 S-DES 可以表示成：

$$C = F(G(F(P,K_1),K_2))$$

交换函数 G 是一个线性操作。对于 S-DES 中的 8 位数据，G 的矩阵形式如下：

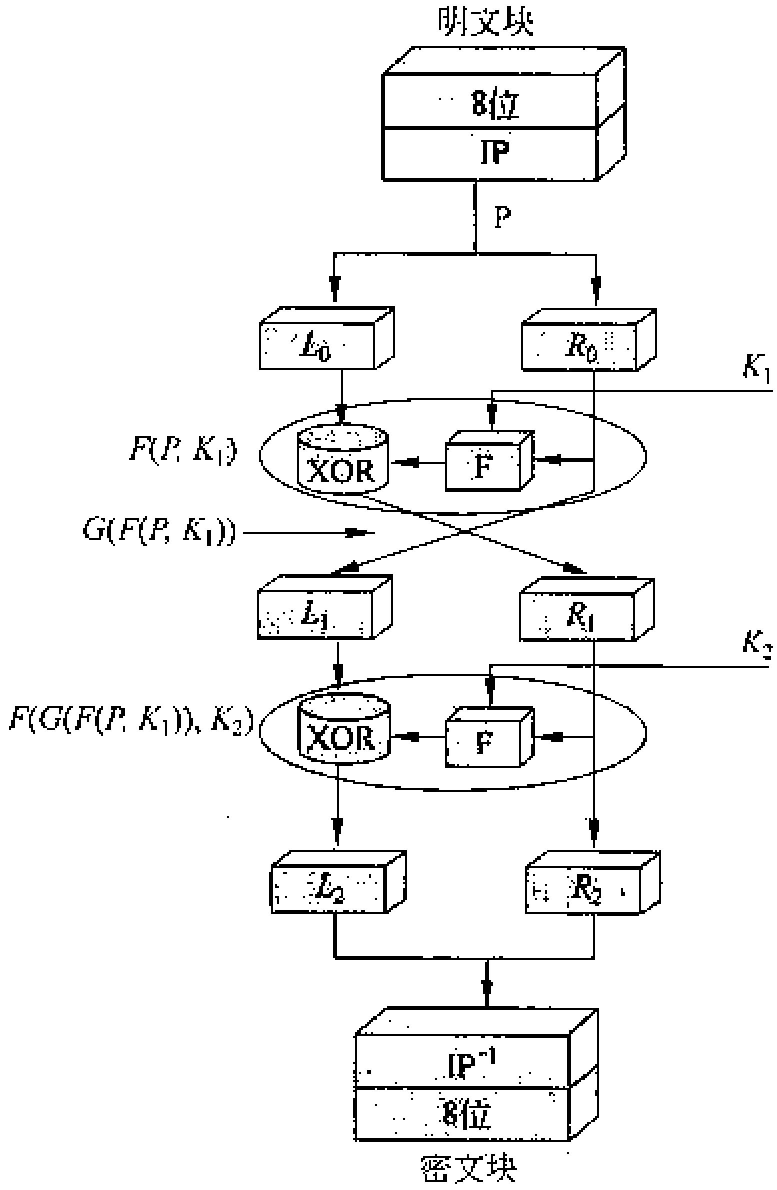


图 7.20 S-DES 的线性密码分析

$$\begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$

如果 F 也是一个线性函数，那么 C 就可以重新写成：

$$C = \mathrm{A}(G(F(P,K_1))) + \mathrm{B}K_2 \pmod 2$$

由于 G 是一个线性函数（$y=Sx$），代入上式得：

$$C = \mathrm{A}[S(\mathrm{A}P + \mathrm{B}K_1)] + \mathrm{B}K_2 \pmod 2$$

展开上式得：

$$C = \mathrm{A}S\mathrm{A}P + \mathrm{A}S\mathrm{B}K_1 + \mathrm{B}K_2 \pmod 2$$

知道了 C 和 P，就可以利用上式求出密钥值，也就是说，这就是明文—密文对攻击法。

现实情况是 F 并不是线性的。因此线性密码分析法的技巧是给出一个较接近的线性函数 F。其真正含义是利用近似的 S 盒来替代块加密法中实际使用的 S 盒，以便于密码分析。

近似 S 盒的线性关系可用来找出密文的位。例如，S-DES 的 S_0 盒将四位的输入 $\{X_1, X_2, X_3, X_4\}$ 转换成两位的输出 $\{Y_1, Y_2\}$。S 盒的线性结构可以如下得出：

$$a_1X_1 \oplus a_2X_2 \oplus a_3X_3 \oplus a_4X_4 \oplus b_1Y_1 \oplus b_2Y_2 = 0$$

对于 a_i、b_i，等于 0 或 1。例如，如果 $a_1=1, a_2=1, a_3=0, a_4=0, b_1=1$ 且 $b_2=0$，那么等式变成了：

$$X_1 \oplus X_2 \oplus Y_1 \doteq 0$$

如果这个等式对 S_0 盒成立，那么它表示有 X_1 XOR $X_2 = Y_1$ 形式的线性关系。如果略大于一半或略小于一半，那么表示 S_0 盒具有近似线性关系。因此，通过查找 a_1、a_2、a_3、a_4、b_1 和 b_2 的值，来得出 S 盒中的线性关系。如图 7.21 所示，前面介绍的关系为 9/16，即略大于一半。通过尝试所有的 a 和 b 值，可以为 S 盒构造一个线性近似表。S-DES 中 S_0 和 S_1 盒的线性近似表如图 7.22 所示。这个表的构造方法是，将每个数减去 8（即 16 的一半），这样做是为了更好地看出某个关系是否是可能的。例如，对于 S_0 盒，关系(4,1)就很可能是线性的(A=4,B=1)，因为它发生的概率是 12/16（12−8=4）这表示等式 $X_2=Y_2$ 成立。对于 S_1 盒，关系(7,0)也很可能，它表示 X_2 XOR X_3 XOR $X_4 = Y_2$ XOR Y_1 成立。

利用这些结果，再加上已知明文—密文对，并在块加密结构中逆向推导回去，就可以找出每个阶段的密钥。这个过程包括最能满足线性关系的密钥值。这种破解方法需要大量的已知明文—密文对，但比差分密码分析法需要的要少。

	X_1	X_2	X_3	X_4	Y_1	Y_2
⇨	0	0	0	0	0	1
	0	0	0	1	1	1
⇨	0	0	1	0	0	0
⇨	0	0	1	1	0	1
⇨	0	1	0	0	1	0
	0	1	0	1	0	0
	0	1	1	0	0	1
⇨	0	1	1	1	1	1
⇨	1	0	0	0	1	0
	1	0	0	1	0	1
	1	0	1	0	0	0
⇨	1	0	1	1	1	1
	1	1	0	0	1	1
⇨	1	1	0	1	0	1
⇨	1	1	1	0	0	1
	1	1	1	1	1	1

图 7.21 S_0 盒的线性关系

S_0

A \ B	0	1	2	3
0	8	−8	−8	−8
1	−8	2	0	−2
2	−8	0	−2	−2
3	−8	0	−4	0
4	−8	0	0	0
5	−8	4	2	2
6	−8	−2	0	−2
7	−8	−2	2	−4
8	−8	0	0	0
9	−8	0	2	−2
A	−8	2	0	2
B	−8	−2	2	−4
C	−8	−2	2	0
D	−8	−2	0	2
E	−8	−4	−2	2
F	−8	0	4	0

S_1

A \ B	0	1	2	3
0	8	−8	−8	−8
1	−8	0	1	1
2	−8	2	−1	−3
3	−8	−2	−1	1
4	−8	2	1	−1
5	−8	−2	1	−3
6	−8	0	−5	3
7	−8	0	3	3
8	−8	0	−1	−1
9	−8	0	−1	−1
A	−8	2	1	−1
B	−8	6	1	3
C	−8	2	3	1
D	−8	−2	3	−3
E	−8	0	1	1
F	−8	0	1	1

图 7.22 S-DES 的线性近似表

7.3.5.3 中间点攻击法

在 7.3.3 节中介绍过，DES 的一种实现是三重 DES。由于它使用的是两个（或三个）DES 密钥和三阶段 DES，因此它比标准的 DES 更安全。你可能想知道双重 DES 如何？为什么不使用两个密钥和两阶段 DES 来获得同样的安全性呢？这样的一种实现如图 7.23 所示。如果强力攻击法攻击标准 DES 需要搜索 2^{56} 个密钥，那么攻击双重 DES 则需要搜索 $2^{56}\times2^{56}=2^{112}$ 个密钥。这可以提供很高的安全等级了。

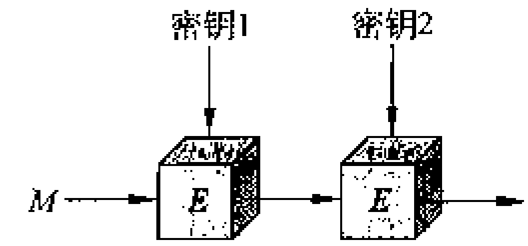


图 7.23 双重 DES 的一种可能实现

但是，这里有一个问题是，如果以存储空间为代价来换取时间消耗的话，那么破解双重 DES 花费的时间只比破解标准 DES 多一点。这个过程需要一个已知明文—密文对，称为**中间点匹配攻击法**(meet-in-the-middle attack)。如果有足够的存储空间可用，用每个可能的密钥将已知明文加密并保存加密结果。然后用每个可能的密钥将密文解密，并将每个结果与内存中的内容进行比较。如果匹配，那么这两个密钥就都找到了。该方法如图 7.24 所示。对一个加密过程，最多需要计算 2^{56} 个密钥，解密过程也是 2^{56} 个密钥，也就是说，只有 $2\times2^{56}=2^{57}$ 个密钥，远少于 2^{112} 个密钥。

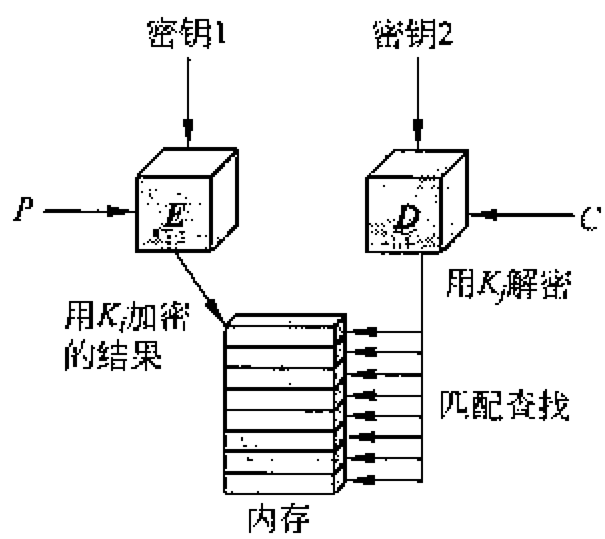


图 7.24 中间点匹配攻击法

例如，假设 Eve 怀疑 Alice 和 Bob 正在使用双重 S-DES 加密法，并且知道 Bob 发送的消息以"Alice"开头。Eve 首先用每个可能的 S-DES 密钥将"Alice"加密。以下是部分密文的列表：

密钥	密文							
0	10110	10110	10001	11001	10110	11110	10111	10100
1	01011	10000	10001	01101	10100	10101	11000	11001
2	00000	10111	01001	10110	10111	11110	00010	00000
⋮				…				
92	11110	10011	10101	10101	10111	01110	10011	10101
⋮				…				
1023	01000	01110	00101	01110	10001	10011	01000	00110

这个表将保存在内存中。当 Eve 截获了一个从 Bob 发送给 Alice 的密文消息后，就用每个可能的密钥将消息的开头部分加密。假设 Eve 截获了如下消息：

10111 10101 00001 11111 01000 01000 01000 00110

利用每个可能的密钥将该消息解密后生成如下的列表：

密钥								解密结果
0	00101	11111	01011	00110	01010	01000	01110	00000
1	10001	11001	11011	10100	00011	11010	00100	00001
2	11100	01001	11101	10100	01010	01100	00001	00111
⋮				…				
122	11110	10011	10101	10101	10111	01110	10011	10101
⋮				…				
1023	11001	11001	10000	11010	10111	11101	11011	00101

通过比较这两个列表，Eve 可以注意到，使用密钥 122 解密的结果与用密钥 92 加密的结果相同，因此，Bob 和 Alice 肯定使用的是密钥对(92,122)。

中间点匹配攻击法不仅限于用来破解双重 DES。事实证明，它还可以用来破解其他加密法，甚至还可以用来破解密钥管理系统(有关内容将在第 9 章介绍)。

7.3.5.4 旁道密码分析

诸如 DES 之类的块加密法有时会在用智能卡中。现在有了一些类似于信用卡的设备，用来维护银行账号或为用户提供强制的身份认证。这些设备中使用的密钥也有方法找出(传统的密码分析法除外)。事实证明，通过观察特定的性能特征，可以发现有关这些加密法操作的信息。这些方法就是所谓的**旁道攻击法**(side-channel attack)。例如，当处理密钥的"1"位时，要消耗更多的能量。通过监控能量的消耗，就可以知道密钥的每个位。还有一种攻击法是监控完成一个算法所耗时间的微秒数。所耗时间数也可以反映出部分密钥的位。

这种攻击法将在第 8 章中更详细地介绍。当你再次遇到它时，请记住旁道密码分析法也是破解块加密法的可行攻击法。

7.3.5.5 Bob、Alice 与 Eve

Bob 和 Alice 还在寻找一种完美的加密法，以永久地防止 Eve 阅读他们的通信。他们对 DES 感兴趣，但觉得密钥太长了，无法记住，因此他们决定使用 S-DES。这也许不是一个理想的选择，但由于某些原因，实际的用户往往就是会做出这样的决策。

没过多久，Eve 就发现了他们使用的新加密法，于是开始寻找破解它的简单方法。她首先是利用 CAP 软件提供的工具进行穷举密钥攻击。

如图 7.25 所示，这种方法能很好地完成工作。

和往常一样，Bob 和 Alice 开始认为他们选择 S-DES 不是明智之举，因此决定升级到双重 S-DES(CAP 提供了这种加密法)。毕竟该加密法有 2^{20} 个密钥，这足以使 Eve 忙一阵子的了。但是，Eve 可以使用中间点匹配攻击法(也是使用 CAP 软件，如图 7.26 所示)。

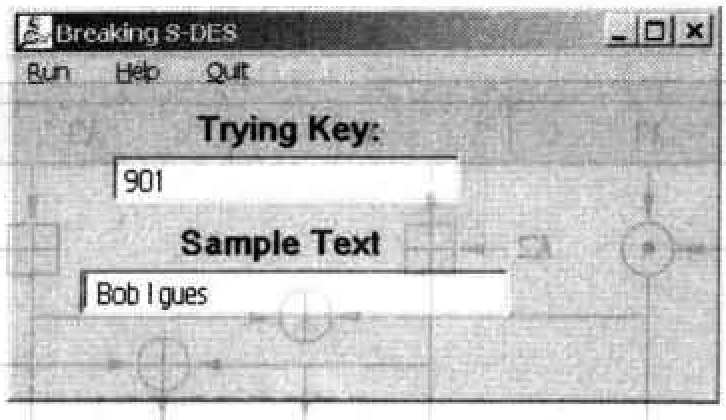


图 7.25 使用 CAP 软件来破解 S-DES

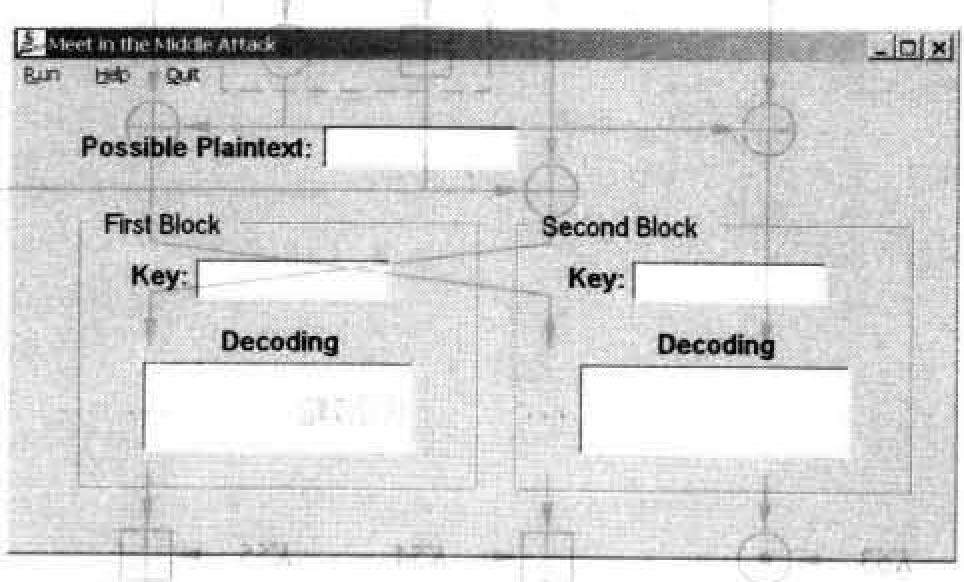


图 7.26 Meet in the Middle Attack 窗口

7.4 IDEA

国际数据加密算法(International Data Encryption Algorithm, IDEA)最先是由 X. Lai、J. L. Massey和 S. Murphy 在 1991 年的一篇论文中公开发表的。它的功能比 DES 更强。现在,它的主要用途是作为内置于 PGP(pretty good privacy)中的一种加密算法。IDEA 使用 128 位的密钥将 64 位明文块加密成密文。但是 IDEA 与 DES 的不同之处不仅仅在于密钥的大小。IDEA 改变了 Feistel 结构,它使用了三种不同的操作:一个 XOR 逻辑运算、两个补加操作和乘积 mod $2^{16}+1$。其中的一些思想将在稍后的高级加密标准(Advanced Encryption Standard, AES)中介绍,见 7.5 节。

IDEA 需要 56 个 16 位的子密钥,这些子密钥是这样生成的:将初始的 128 位密钥连续滚动 25 位。也就是说,初始的 128 位密钥 K,先分成最初的 8 个 16 位的子密钥,然后,将 K 左移 25 位并分成接下来的 8 个 16 位密钥。反复进行该过程,直到所有的 56 个子密钥全部完成。

明文分成 4 个 16 位的数,每段对应于 IDEA 的 8 轮和最后的转换轮。每轮中包含有三种操作,如图 7.27 所示。

M-A(乘-加)盒是 IDEA 真正的加密之处。每个输入位与所有输入位都相关。IDEA 可以抗差分和线性密码分析法的攻击。

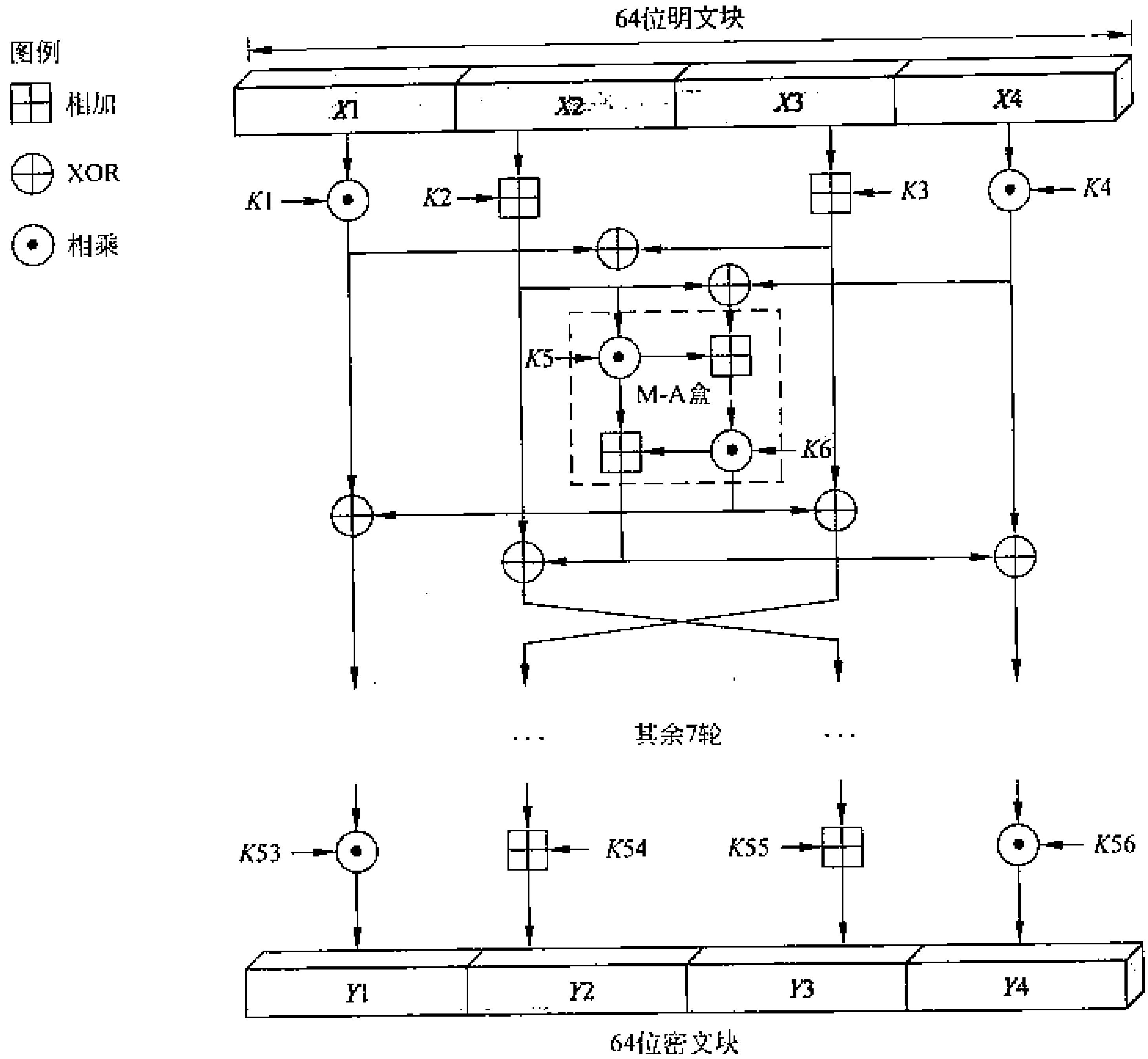


图 7.27 IDEA

7.5 高级加密标准

随着新的密码分析技术的开发，DES 变得不安全了，于是美国国家标准与技术局(National Institute of Standards and Technology，NIST)于 1999 年发出了一个通告，要求开发新的加密标准。其要求如下：

- 应是对称块加密法，具有可变长度的密钥(128、192 或 256 位)，一个 128 位的块。
- 应比三重 DES 更安全。
- 应可应用于公共领域并免费提供。
- 应至少在 30 年内是安全的。

总共有来自 10 个不同国家的 15 种算法。在选定最终算法之前，这些算法先经受了一个很长的公开分析过程。

NIST 使用了一种比较正确的方式来选定块加密法标准。它要求公开算法以及来自加密社区的评论。在三个公开的会议上，展示并讨论这些算法，并尽量暴露它们的弱点。第二次会议之后，15 种算法就剩下 5 个候选者：IBM 提交的 MARS；RSA 实验室提交的 RC6；Joan Daemen 和 Vincent Rijment 提交的 Rijndael；Anderson、Biham 和 Knudsen 提交的

Serpent；Schneier、Kelsy、Whiting、Wagner、Hall 和 Ferguson 提交的 Twofish。这 5 个优秀的算法都经受了 6 个月的考验，然后再次察看，又经过 6 个月的测试后，NIST 宣布了最终的获胜者，这就是 Rijndael(读作“rain-doll”)。

7.5.1 Rijndael 结构

Rijndael 是一种灵活的算法，其块的大小可变(128、192 或 256 位)，密钥大小可变(128、192 或 256 位)，迭代次数也可变(10、12 或 14)，而且迭代次数与块和密钥大小有关。正因为其灵活，Rijndael 实际上有三个版本：AES-128、AES-192 和 AES-256。常见的 Rijndael 结构如图 7.28 所示。Rijndael 不像 DES 那样在每个阶段中使用替换和置换，而是进行多重循环的替换、列混合密钥加操作。从这来看，Rijndael 很像 IDEA，没有进行包含置换操作的典型 Feistel 轮(注意，这里把术语 AES 和 Rijndael 视为等价的了，可交替使用)。

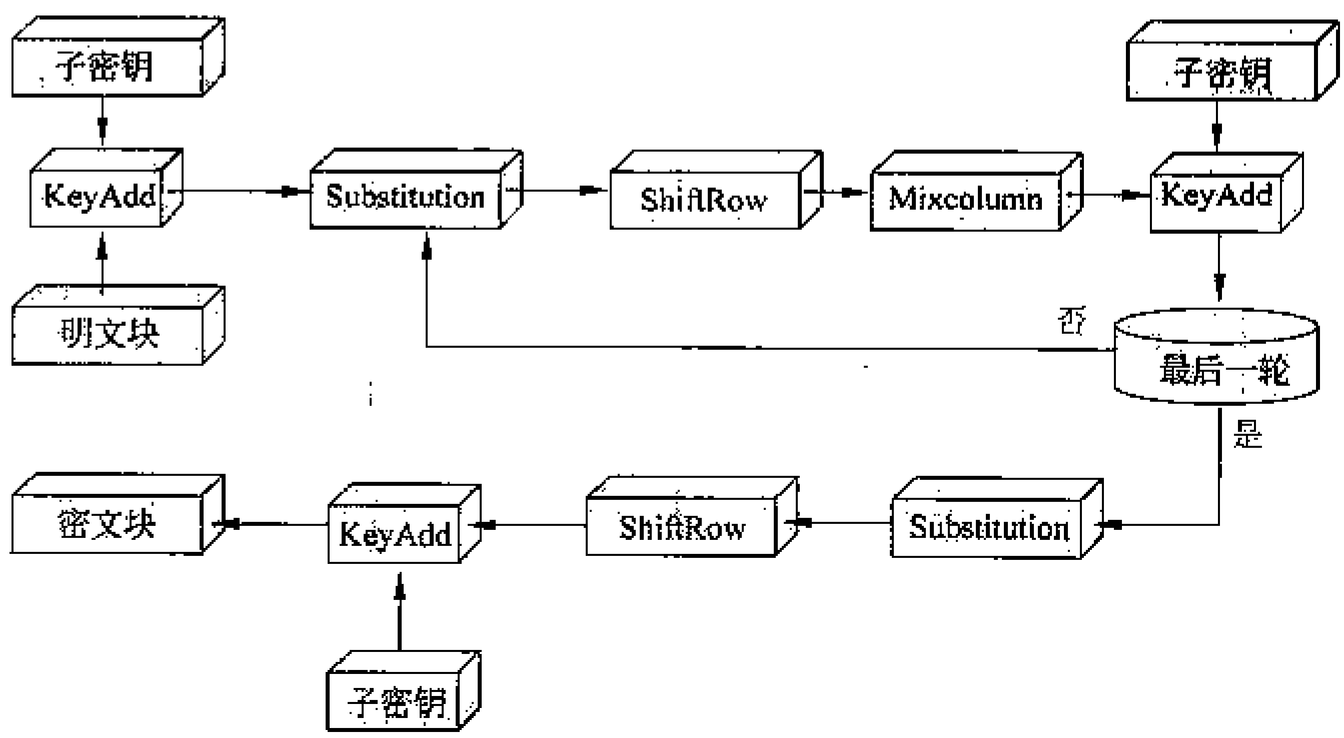


图 7.28 常见的 Rijndael 结构

Rijndael 首先将明文按字节分成列组。前 4 个字节组成第一列，接下来的 4 个字节组成第二列，以此类推，如图 7.29 所示。如果块为 128 位，那么就可组成一个 4×4 的矩阵。

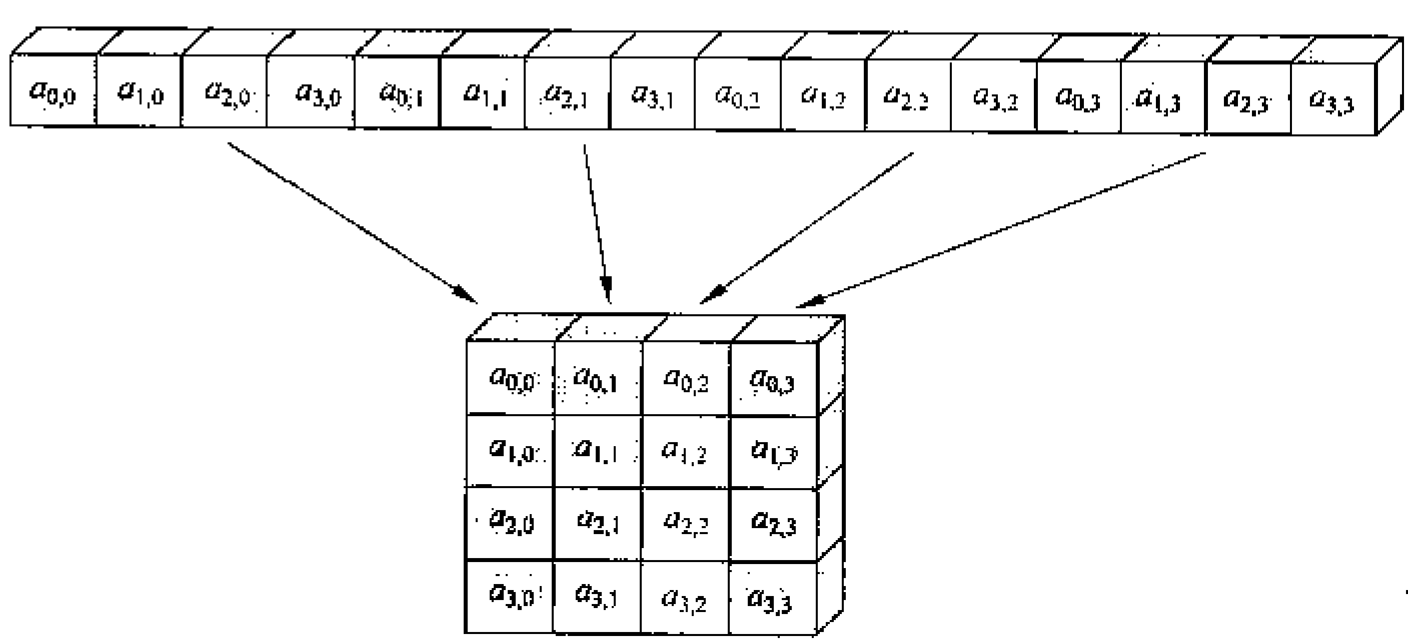


图 7.29 Rijndael 的分组

对于更大的块，矩阵的列相应地增加。用相同方法也将密钥分成矩阵。Rijndael 替换操作使用的是一个 S 盒（而不是像 DES 那样使用 8 个）。Rijndael 的 S 盒是一个 16×16 的矩阵，如图 7.30 所示。列的每个元素作为输入用来指定 S 盒的地址：前 4 位指定 S 盒的行，后 4 位指定 S 盒的列。由行和列所确定的 S 盒位置的元素取代了明文矩阵中相应位置的元素。

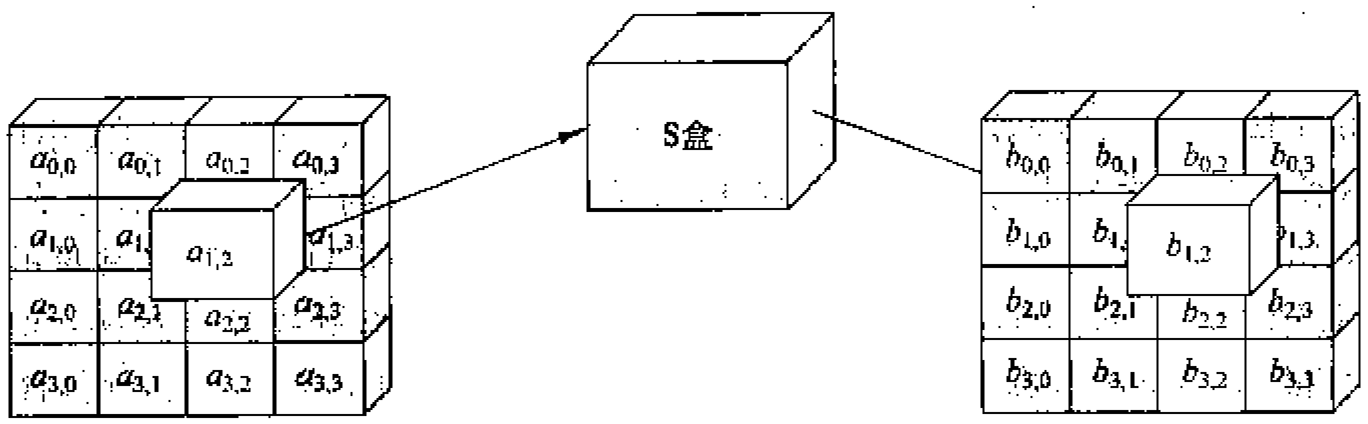


图 7.30 Rijndael 的替换操作

Rijndael 的 S 盒实际上是执行从输入到输出的代数转换。其矩阵表示形式如下：

$$\begin{vmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{vmatrix} \begin{vmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{vmatrix} + \begin{vmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{vmatrix}$$

字节 a 与给定的矩阵相乘，其结果再加上固定的向量值 63（用二进制表示）。这可以用图 7.31 所示的 S 盒格式来表示。例如，输入矩阵（用十六进制表示）与相应的输出如下所示：

输入				输出			
12	2a	21	0b	c9	e5	fd	2b
45	bd	04	c1	6e	7a	f2	78
23	0a	00	1c	26	67	63	9c
89	11	2a	fc	a7	82	e5	b0

在上面的示例中，第一项为 12，它将被 S 盒中行 1 列 2 的元素 c9 取代。其余的输出也可用相同的方法确定。这种 S 盒的设计是有一定的限制条件的，其中的一些限制条件是：它应是可逆的，不能有固定点（也就是说，不能存在这样的情况：行 i 列 j 位置上的值为 ij）。要描述 S 盒简单，但它其实是一个复杂的代数结构（你相信吗？）。

行移位操作是作用于 S 盒的输出的，其中，列的 4 个行螺旋地左移，即第一行左移 0 位，第二行左移 1 位，第三行左移 2 位，第四行左移 3 位，如图 7.32 所示。从该图中可以看出，这使得列完全进行了重排，即在移位后的每列中，都包含有未移位前每个列的一个字节。接下来就可以进行列内混合了。

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

图 7.31　Rijndael 的 S 盒

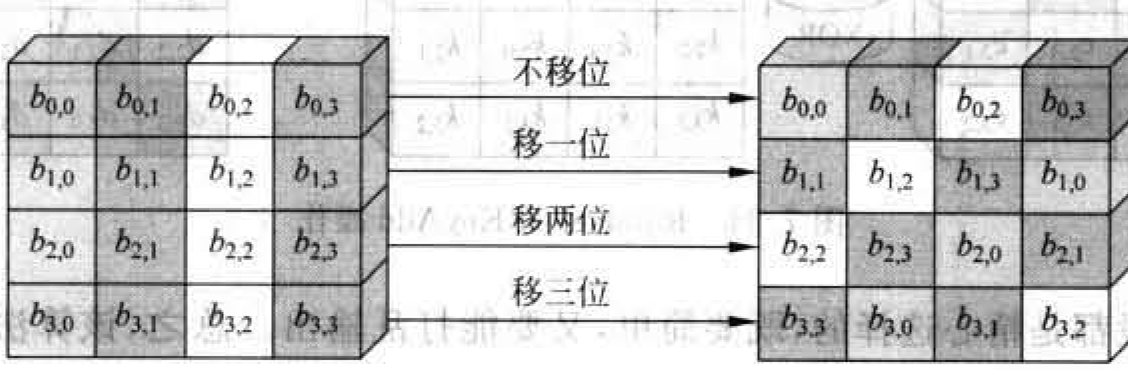


图 7.32　Rijndael 移位操作

列混合是通过矩阵相乘来实现的。经移位后的矩阵与固定的矩阵(以十六进制表示)相乘，如下所示：

$$\begin{vmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{vmatrix} = \begin{vmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 01 & 03 \\ 03 & 01 & 01 & 02 \end{vmatrix} \begin{vmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{vmatrix}$$

其图形表示见图 7.33 所示。例如，如果第一列为 11、09、01 和 35(以十六进制表示)，那么乘操作为：

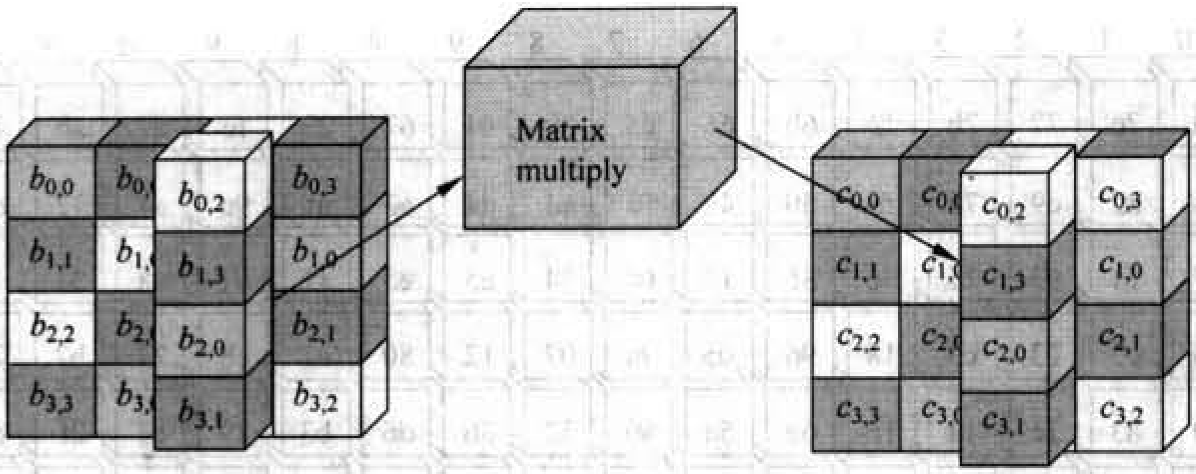


图 7.33 Rijndael 的列混合操作

$$\begin{vmatrix}73\\6b\\ba\\a7\end{vmatrix}=\begin{vmatrix}02&03&01&01\\01&02&03&01\\01&01&02&03\\03&01&01&02\end{vmatrix}\begin{Vmatrix}11\\09\\01\\35\end{Vmatrix}$$

因此,新的第一列包含了字节 73、6b、ba 和 a7。这些操作保证了明文位经几个迭代轮后已高度打乱了,同时还保证输入与输出之间的关联很少了。这就是该算法安全性的两个重要特性。解密操作使用的是不同的矩阵。

最后一个阶段是将以上结果与子密钥进行 XOR 逻辑运算(注意,该子密钥是从初始密钥派生而来的),如图 7.34 所示。这完成了该算法的一次迭代。

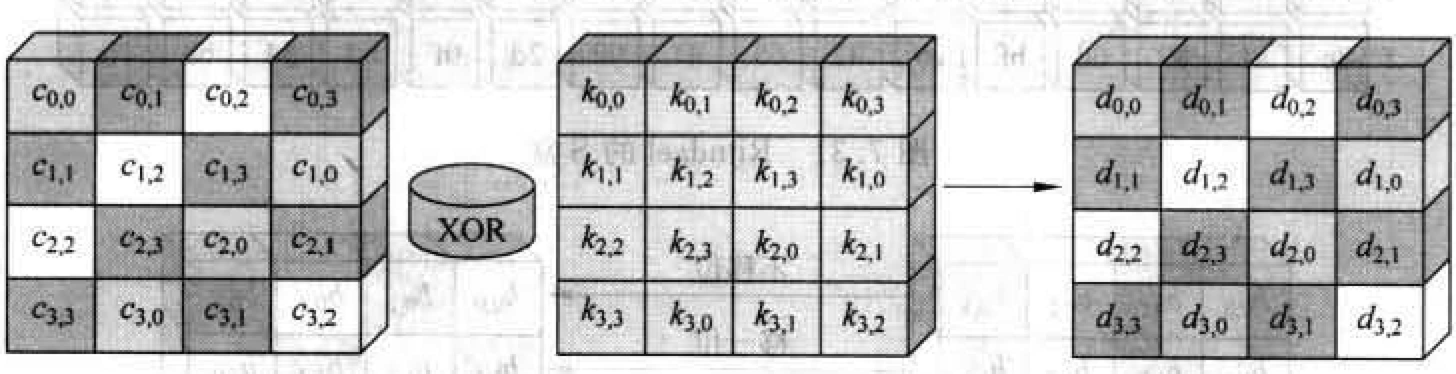


图 7.34 Rijndael 的 KeyAdd 操作

以上各阶段都是精心选择的,既要简单,又要能打乱输出。总之,该算法完成了一项令人惊奇的工作。

现在让我们来跟踪 AES 的每一个迭代,以观察所有操作对输出的影响。假设 Alice 要给 Bob 发送这样一个短消息:"Bob look at this"。包括空格在内,这个消息正好为 16 个字符(表示成 ASCII 码为 128 位)。我们不用去管每个位是什么(你不会去阅读这 128 个位,我也不想在这里将它们写出来),而是看看以十六进制为单位的字节在 AES 操作中的变化情况。这个消息表示成十六进制为:

42 6f 62 20 6c 6f 6f 6b 20 61 74 20 74 68 69 73

写成 4×4 的矩阵形式为:

42	6c	20	74
6f	6f	61	68
62	6f	74	69
20	6b	20	73

该矩阵作为S盒的输入。第一个输入为42，它指定了S盒中行为4列为2的单元，其内容为2c。在S盒中查找出每个元素，从而生成如下的输出矩阵：

2c	50	b7	92
a8	a8	ef	45
aa	a8	92	f9
b7	7f	b7	8f

这种替换实现了AES的第一次打乱。接下来的一个阶段是旋转各行：

2c	50	b7	92
a8	ef	45	a8
92	f9	aa	a8
8f	b7	7f	b7

该操作通过混淆行的顺序来实现AES的第一次扩散。接下来的一个阶段是进行乘法操作。对于上面的示例，第一列进行如下形式的转换：

$$\begin{vmatrix} 72 \\ d1 \\ ad \\ 66 \end{vmatrix} = \begin{vmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 01 & 03 \\ 03 & 01 & 01 & 02 \end{vmatrix} \begin{vmatrix} 2c \\ a8 \\ 92 \\ 8f \end{vmatrix}$$

固定矩阵乘以每列后得出：

72	19	66	4b
d1	d1	be	91
ad	d0	d0	07
66	46	23	74

然后就要用到子密钥了。在我们的示例中，子密钥为：

01	a3	90	12
e1	44	20	11
cc	73	04	a9
59	06	30	b4

将矩阵相乘的结果与子密钥进行XOR逻辑运算，可得：

72	19	66	4b		01	a3	90	12		73	ba	f3	59
d1	d1	be	91	XOR	e1	44	20	11	=	30	95	9e	80
ad	d0	d0	07		cc	73	04	a9		61	a0	d4	a6
66	46	23	74		59	06	30	b4		3f	40	13	c0

将上面第一轮得到的输出与初始输入进行比较：

42 6f 62 20 6c 6f 6f 6b 20 61 74 20 74 68 69 73

73 30 61 3f ba 95 a0 40 f3 9e d4 13 59 80 a6 c0

如果转换成二进制位，可以发现在全部128个位中有76个位发生了改变，而这仅仅是一轮，还要进行另外10轮。

7.5.2 密钥生成

密钥是按矩阵的列进行分组的，然后添加 40 个新列来进行扩充。如果前 4 列（即由密钥给定的那些列）为 $W(0)$、$W(1)$、$W(2)$和 $W(3)$，那么新列以递归方式产生。

如果 i 不是 4 的倍数，那么第 i 列由如下等式确定：

$$W(i) = W(i-4) \text{ XOR } W(i-1)$$

如果 i 不是 4 的倍数，那么第 i 列由如下等式确定：

$$W(i) = W(i-4) \text{ XOR } T[W(i-1)]$$

其中 $T[W(i-1)]$是 $W(i-1)$的一种转换形式，按以下方式实现的：

(1) 循环地将 $W(i-1)$的元素移位，每次一个字节，也就是说，abcd 变成了 bcda。

(2) 将这 4 个字节作为 S 盒的输入，输出新的 4 个字节 efgh。

(3) 计算一轮的常量 $r(i)=2^{(i-4)/4}$。

(4) 这样生成转换后的列：$[e \text{ XOR } r(i), f, g, h]$。

第 i 轮的轮密钥组成了列 $W(4i)$、$W(4i+1)$、$W(4i+2)$和 $W(4i+3)$。

该过程如图 7.35 所示。例如，如果初始的 128 位密钥为（以十六进制表示）：

3ca10b21　57f01916　902e1380　acc107bd

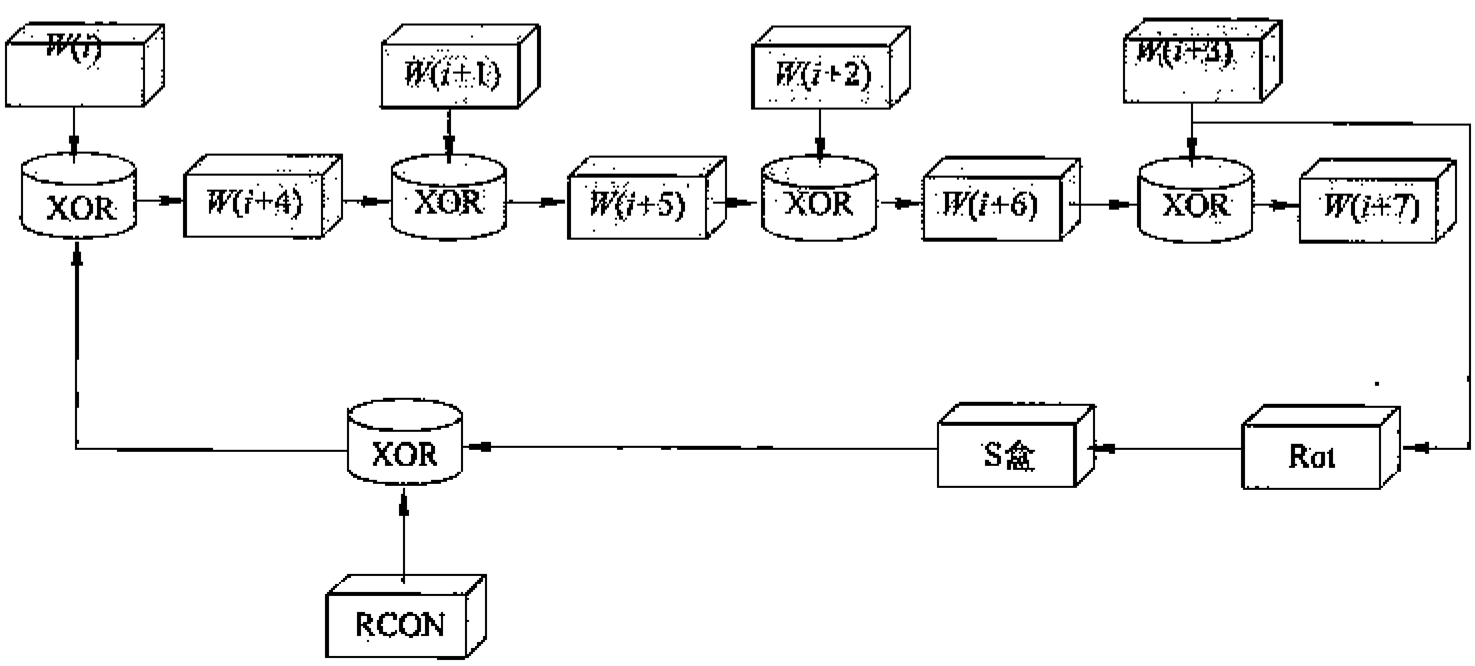


图 7.35　Rijndael 的密钥生成

那么 4 个初始值为 $W(0)=$3ca10b21，$W(1)=$57f01916，$W(2)=$902e1380，$W(3)=$acc107bd。下一个子密钥段为 $W(4)$，由于 4 是 4 的倍数，所以：

$$W(4) = W(0) \text{ XOR } T[W(3)]$$

$T[W(3)]$的计算步骤如下：

(1) 循环地将 $W(3)$的元素移位：acc107bd 变成了 c107bdac。

(2) 将 c107bdac 作为 S 盒的输入，输出为 78 85 7a 91。

(3) 计算一轮的常量 $r(4)=2^0=01$（以十六进制表示）。

(4) 将 $r(4)$与第一个字节 78 进行 XOR 逻辑运算：78 XOR 01=79。

因此，$T[W(3)]=$79857a91，并且

$$W(4) = \text{3ca10b21 XOR 79857a91} = \text{452471b0}$$

其余的三个子密钥段计算如下：

$$W(5) = W(1)\ \text{XOR}\ W(4) = \text{57f01915 XOR 452472b0} = \text{12d468a5}$$

$$W(6) = W(2)\ \text{XOR}\ W(5) = \text{902e1380 XOR 12d468a5} = \text{82fa7b25}$$

$$W(7) = W(3)\ \text{XOR}\ W(6) = \text{acc107bd XOR 82fa7b25} = \text{2e3b7c98}$$

于是，第一轮的密钥为452471b012d468a582fa7b252e3b7c98。

7.5.3 AES的操作

AES(Rijndael)有取代当前所有应用中使用的DES的趋势。它同时还必须有足够的安全性来取代三重DES应用。与DES一样，AES也可用于5种块加密法模式：CBC、ECB、OFB、CTR和OCB。另外，有人还曾经提议将AES作为一种流加密模式(称为**整数计数器模式**，integer counter mode，ICM)用于一种新的块加密法中。ICM模式如图7.36所示。它使用AES(或其他块加密法)将128位的初始值加密成一个128位的流。然后增加初始值并生成另一个128位的流。重复该过程，直到生成了足够的位。

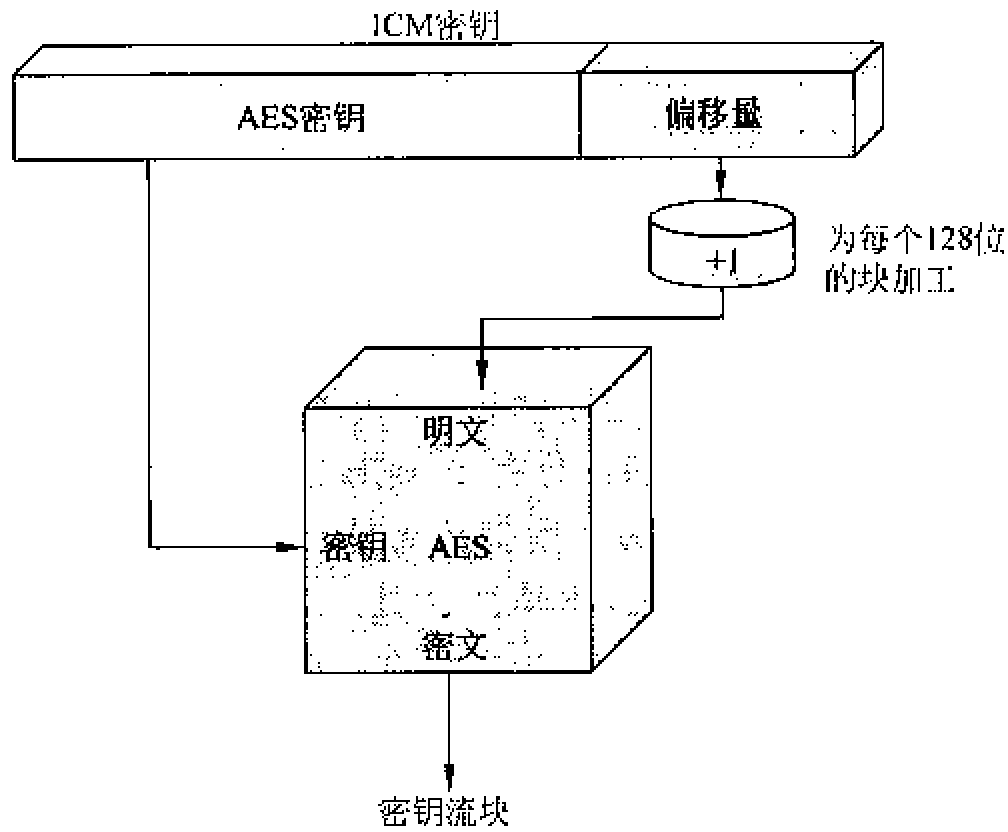


图7.36 AES的ICM模式

由于Rijndael非常小，运行速度快，密钥大小128位，安全可靠，因此，它不仅可用于安全要求比三重DES更高的应用中，而且可用于计算资源有限的应用中。这就是说，在手持和无线计算市场，它很有应用潜力。

7.5.4 AES的安全性

在AES参与竞选的过程中，所有提交的算法都告知了密码分析员。在那个相对较短的时间里，谁也无法提出一种可破解Rijndael(或其他候选算法)的技术。因此，看上去Rijndael是安全的。但是，随着AES的采用，密码分析员还在继续研究该算法以找出其弱

点。他们已取得了不同程度的成功。

AES 可以抗差分和线性密码分析法的攻击。这并不奇怪,因为 Rijndael 在设计时就考虑了这些攻击法。但是,已经出现了其他一些攻击法,可以成功破解轮数少的 AES。这些攻击法是差分和线性密码分析法的变体。**不可能差分**(impossible differential,ID)攻击法已成功破解了 6 轮的 AES-128。**平方**(square)攻击法已成功破解了 7 轮的 AES-128 和 AES-192。**冲突**(collision)攻击法也已成功破解了 7 轮的 AES-128 和 AES-192。所有这些攻击法对于全部 10 轮的 AES-128 都失败了,但这表明,AES 可能有有待发现的弱点。

2003 年 1 月,因特网上发表了一篇文章,说可能已发现了这样一个有待发现的弱点。该文章说,在 AES 的输出位中发现了一种趋势,可以用称为**明文相关重复代码**(plaintext-dependent repetition-code)攻击法来破解。正如所预期的那样,这篇文章在密码分析界引起了相当的反响。但没过多久就有人反映,认为这种攻击法对 AES 无效,在 AES 的输出位中也不存在这种趋势。尽管在这篇文章中存在某些基本的错误,但这个过程说明,发现和验证 AES 的弱点的努力仍在进行着。

7.5.5 硬件实现

现代块加密法(如 DES 和 AES)既可以用硬件也可以用软件来实现。尽管软件实现更灵活,但越来越多的应用要求用硬件实现。AES 当年能胜出,一部分原因就是它能用软件和硬件有效地实现。

加密算法用硬件实现主要有两个原因。第一个原因是,对很多应用来说,软件实现太慢了。随着块加密法越来越复杂,情况更是如此。对块加密法来说,现在的软件加密速率不超过 100Mbps,对公开密钥算法(有关内容将在下一章中介绍),就更慢了。第二个原因是硬件实现更安全。算法(也许和密钥一起)固化在硬件中,这使得比软件更不易修改。

另一个考虑的因素是,硬件实现的速度优势可能会暴露该加密法的一个弱点,而这用相比较更慢的软件实现中则不明显或不易被发现。例如,如果设计一台基于 DES 硬件芯片的计算机,DES 并不可靠。硬件实现很重要的一点是,可以不断地监控块加密法的整体安全。

通常,将 AES(或其他块加密法)嵌入到硬件有两种方法。一种方法称为 ASIC 实现,另一种称为 FPGA 实现。ASIC 意为**应用特定集成电路**(application specific integrated circuit),这是一种在硅片上进行固化实现的方法,加密算法转换成硬件设计并组装在一个芯片上。FPGA 意为**现场可编程门阵列**(field programmable gate array)。在这两种方法中,FPGA 更为灵活。FPGA 由预先设计好的阵列芯片以及用可编程线路连接起来的可编程逻辑元素组成。

要实现快速的加密法,通常推荐使用自定制设计的 ASIC 方法。在 ASIC 方法中,其加密法算法转换成硬件实现并置于一个硅片上。Verbauwhede、Schaumont 和 Kuo 于 2003 年设计并构造了一个这样的芯片。整个芯片的结构如图 7.37 所示。

输入输出为 16 位的块。这是因为 AES-256 需要 256 个输入针脚,而 ASIC 芯片没有足够的针脚。因此,设计成输入块以节省输入针脚,从而满足 128、196 和 256 位的要求。密钥生成单元负责完成用硬件确定子密钥的全部计算工作。加密单元在一个时钟周期内实现

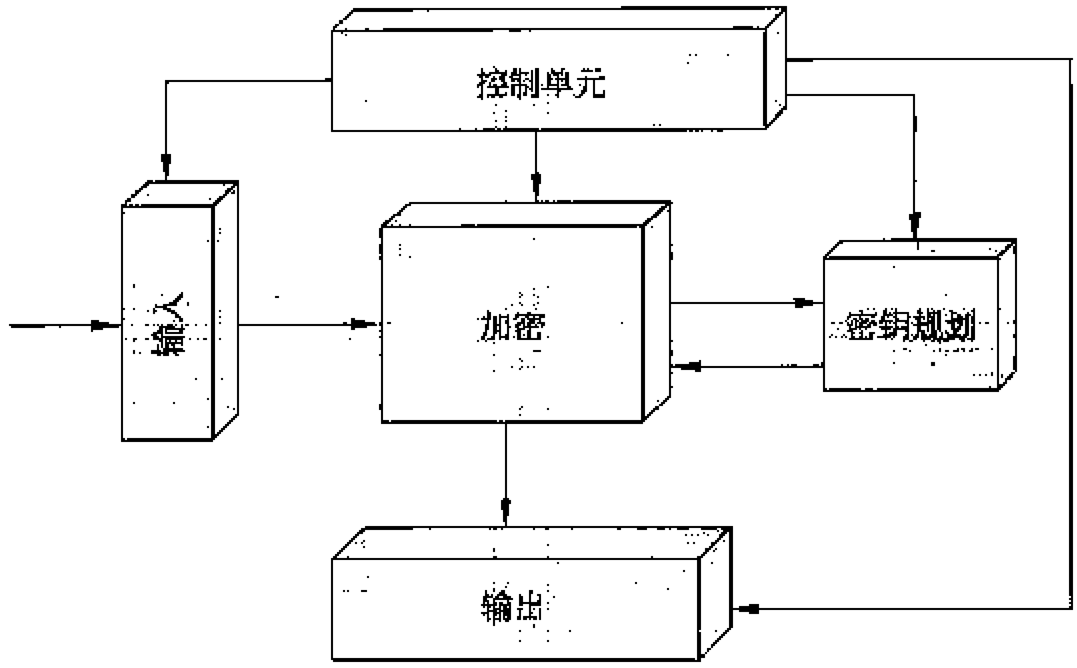


图 7.37 AES 的硬件实现结构

AES 加密的一轮。为了实现这些，这就必须并行的操作数据。加密单元在一个时钟周期内实现 AES 的一轮。另一种方法是使用含有 32 个 S 盒的硬件。

AES-128 的硬件加密如图 7.38 所示。注意，每 16 个位输入就有一个 S 盒，并且移位操作是由连接方法来实现的。有 4 个矩阵乘法单元，它与 16 个 S 盒一起，实现了芯片的并行计算能力。这种特色的芯片由 173000 个逻辑门组成，运行频率为 125MHz，吞吐量为 2.29GBps。这不是 ASIC 实现的惟一示例。还有不少其他方法见诸于文字报道中，但很少在现代的商业应用上可行。

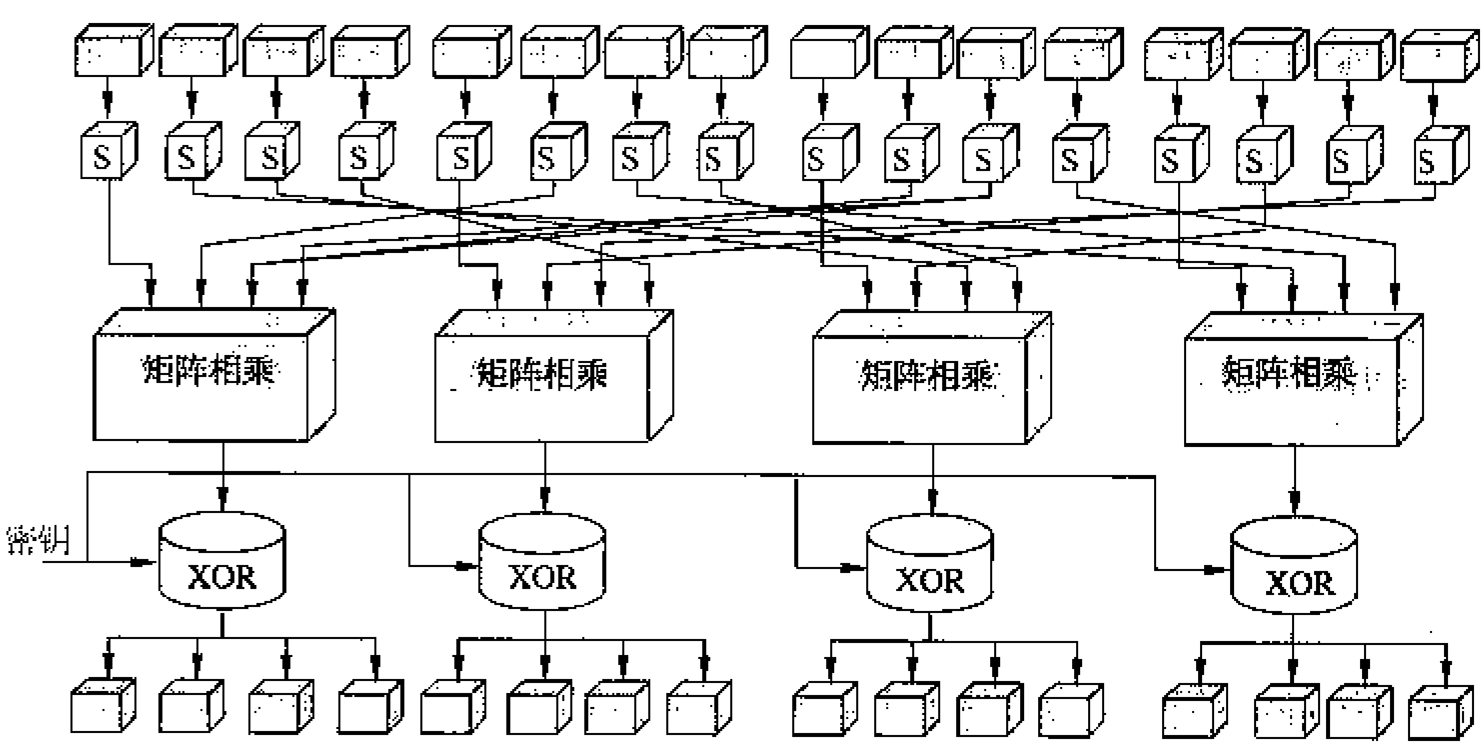


图 7.38 AES 加密硬件实现

正如前面已介绍的那样，一种替代 ASIC 实现的方式是现场可编程门阵列实现。FPGA 的逻辑块集是预先设计好的，这些逻辑块相互连接成阵列，如图 7.39 所示。每个逻辑块由复杂的电路元件集组成。这些电路元件可实现各种的布尔函数。它们还有一定的记忆能力。这些块是"可编程的"，即可设置或清除其内部连接，从而实现块的特殊逻辑结构。块之间的线路允许信号沿可编程链接发送。也就是说，块的输入输出可通过编程来连接高某个线路或跳过这个线路。该线路由可切换矩阵从垂直方向切换到水平方向。

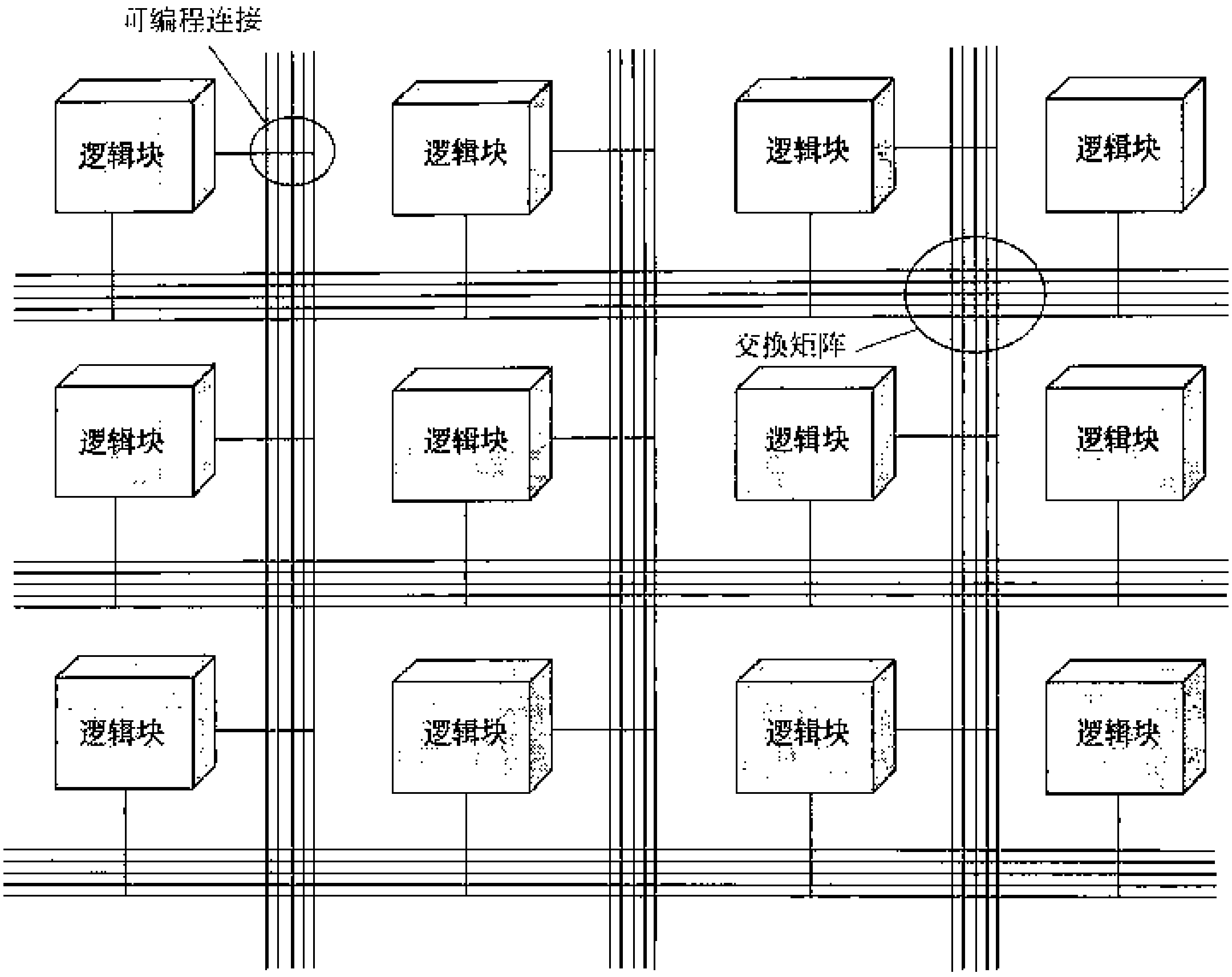


图 7.39 典型的 FPGA 结构

以上只是该复杂设备的一个简单介绍，可以把它看作是一个系统，它具有灵活的硬件结构，可通过"编程"来实现某个特殊的功能，而不是用固化的硬件结构来运行程序指令集。有好几个制造商提供了多种 FPGA 和技术。商用的 FPGA 具有超过二百万个的逻辑门。

作为评估 AES 的一部分，不少大学的研究人员设计了 Rijndael 的 FPGA 实现。设计的速度从 294MBps 到 414MBps 不等。一个 FPGA 实现比软件实现更快，但比自定制的 ASIC 芯片更慢。但是，FPGA 比 ASIC 更容易设计，也更便宜。

7.5.6 其他候选加密算法

选择 Rijndael 作为新的 AES 有很多个原因。它并不是最安全、运行最快或实现结构最小的加密算法，但 NIST 认为总体上 Rijndael 是最好的选择。每个候选的加密算法都不是"失败者"。它们都提供了一些有趣的思想，都有取代 DES 的价值。由于在竞争期间都对它们进行详细的研究，所以知道了它们的功能和弱点。显然，其他的候选加密算法还将会继续使用。因此，这里有必要简单地介绍它们的一些特性。

7.5.6.1 RC6

RC6 是 AES 的候选者之一，由 Rivest、Robshaw、Sidney 和 Yin 提交的。它可能是最简单的 AES 算法，它作用于包含有 128 位的明文块，使用的密钥大小为 128、192 或 256 位，共进行 20 轮的加密。

RC6 需要 44 个密钥，这些密钥是利用一系列复杂的运算从初始的 128 位密钥生成而来的。初始密钥分成四个单词，并放在数组 $L(0 \cdots 3)$中。这 44 个子密钥在数组 $S(0 \cdots 43)$中进行初始化，使用的是两个自然常数，一个是 e(自然对数的基数)，另一个是 Φ(黄金分割率)。具体过程如下：

S[0] = b7e1519B (十六进制)

For i = 1 to 43, do

S[i] : = S[$i-1$] × 9e3779B9 (十六进制)

$A = B = i = j = 0$

For k = 1 to 132, do

$A = S(i) = [S(i) + A + B] <<< 3$

$A = L(j) = [L(j) + A + B] <<< (A+B)$

$i = (i+1)$ mod 44

$j = (j+1)$ mod 44

一旦有了子密钥数组 S，就可以进行真正的加密了，首先是将 128 位的明文分成 4 个 32 位的块，分别标记为 A、B、C 和 D。块 B 和 D 进行如下修改：

$$B = B + S(0)$$

且

$$D = D + S(1)$$

接下来是 20 轮的加密操作，包括 XOR 逻辑运算、乘法、移位和加法。第 i 轮如图 7.40 所示。F 盒执行一个乘法和加法操作，具体定义如下：

$$t = [B \times (2B+1)]$$

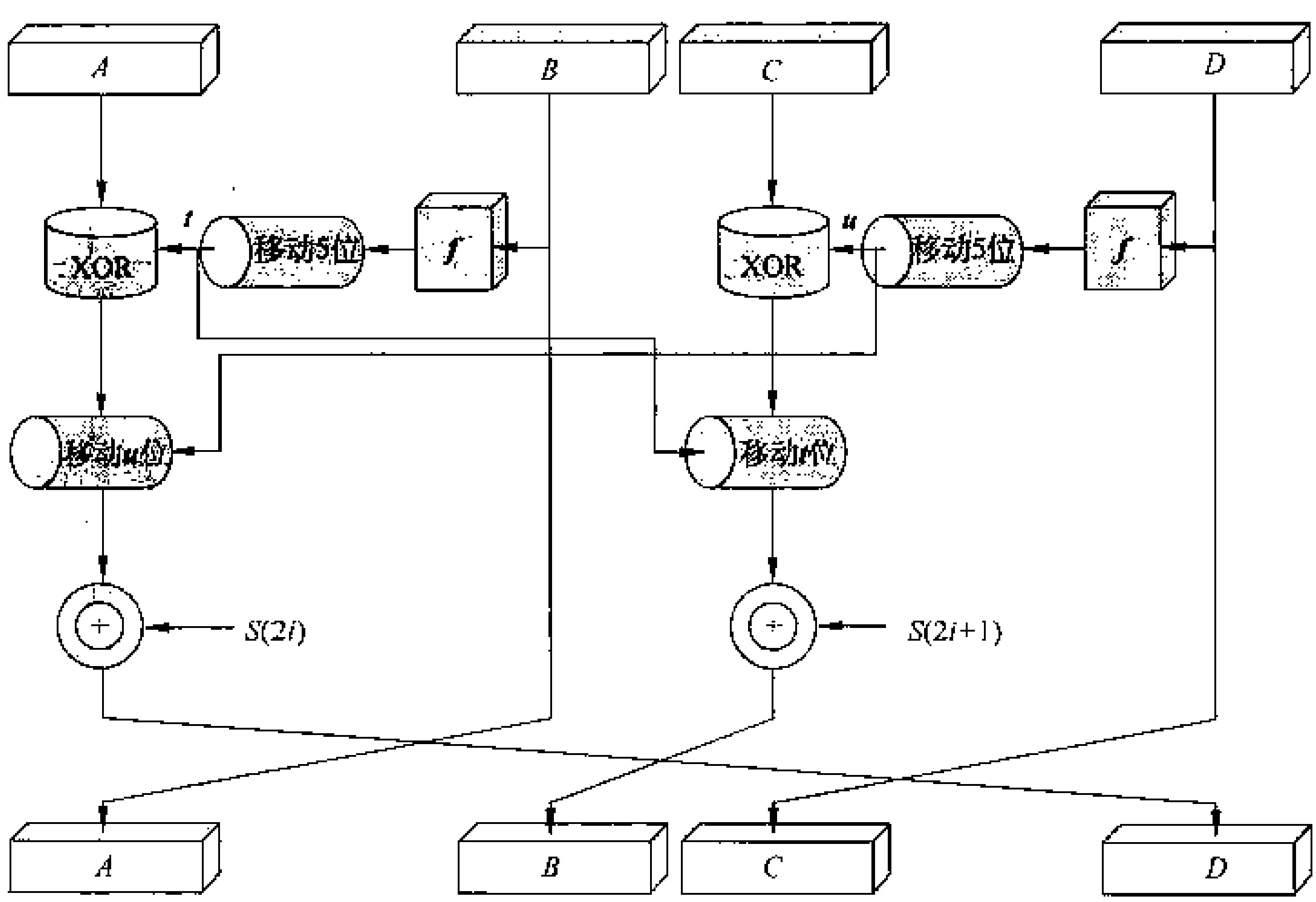


图 7.40 RC6 的一个加密轮

且

$$u = [D \times (2D+1)]$$

20 轮后，块 A 和块 C 在最后一个阶段中进行如下的修改：

$$A = A + S(42)$$

且

$$C = C + S(43)$$

这个过程在加密法的开始和结束处都使用了一些密钥。事实表明，这增加了密钥查找攻击的难度，部分原因是它隐藏了每轮中的实际输入和输出。

7.5.6.2　Twofish

Twofish 是另 AES 的另一个候选者，由 Schneier 提交，它同时具有 RC6 和 Rijndael 的某些特性。它作用于 128 位的块，使用多达 256 位的可变长度密钥。它使用有 16 轮的 Feistel 结构来加密明文，应用了一些特殊的操作，例如与密钥相关的 S 盒和矩阵相乘操作等。

Twofish 使用一个非常复杂的密钥生成算法来生成 40 个子密钥（$K_0 \sim K_{39}$）。

子密钥既用于 Feistel 结构各轮的 F 盒中，也用于生成与密钥相关的 S 盒。S 盒的这种动态特性提供了这样一种特点：攻击者无法预先分析它们。

Twofish 首先将明文分成 4 个 32 位的块，每个块与一个子密钥进行 XOR 逻辑运算（块 A 与 $K(0)$进行 XOR 逻辑运算，块 B 与 $K(1)$进行 XOR 逻辑运算，块 C 与 $K(2)$进行 XOR 逻辑运算，块 D 与 $K(3)$进行 XOR 逻辑运算。）。完成最后一轮的操作后，每个块与 $K(4) \sim K(7)$子密钥进行 XOR 逻辑运算。所有 16 轮都进行了一个 XOR 操作，单个位的移位，一个特殊的函数 f，以及密钥生成。其中的单个轮如图 7.41 所示。

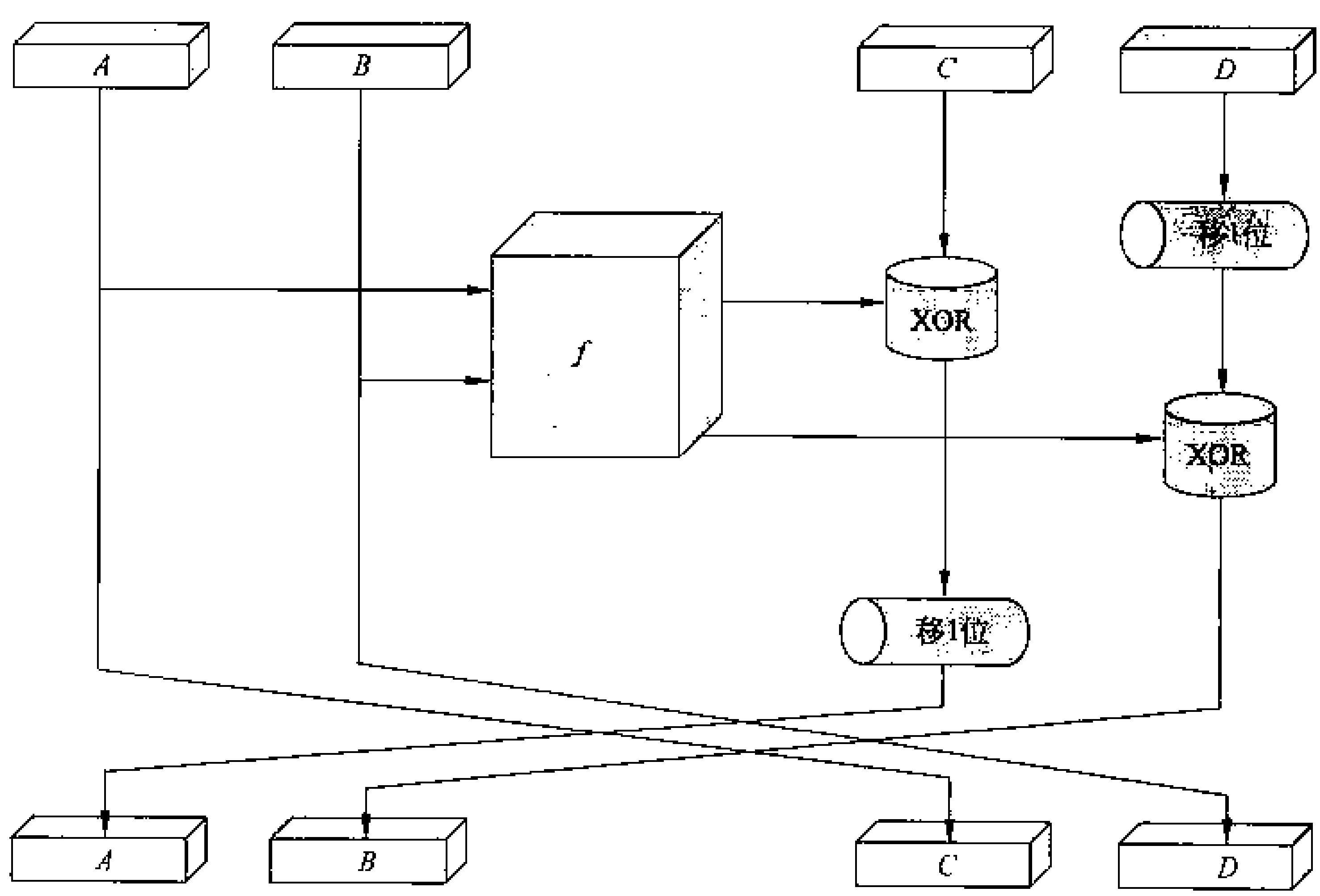


图 7.41　Twofish 的单个轮

函数 f 是由多个 S 盒、一个矩阵乘操作以及与子密钥的加操作组成的，如图 7.42 所示。每个 S 盒接收 8 位的输入，生成 8 位的输出。输出与一个 4×4 的 MDS 矩阵相乘。

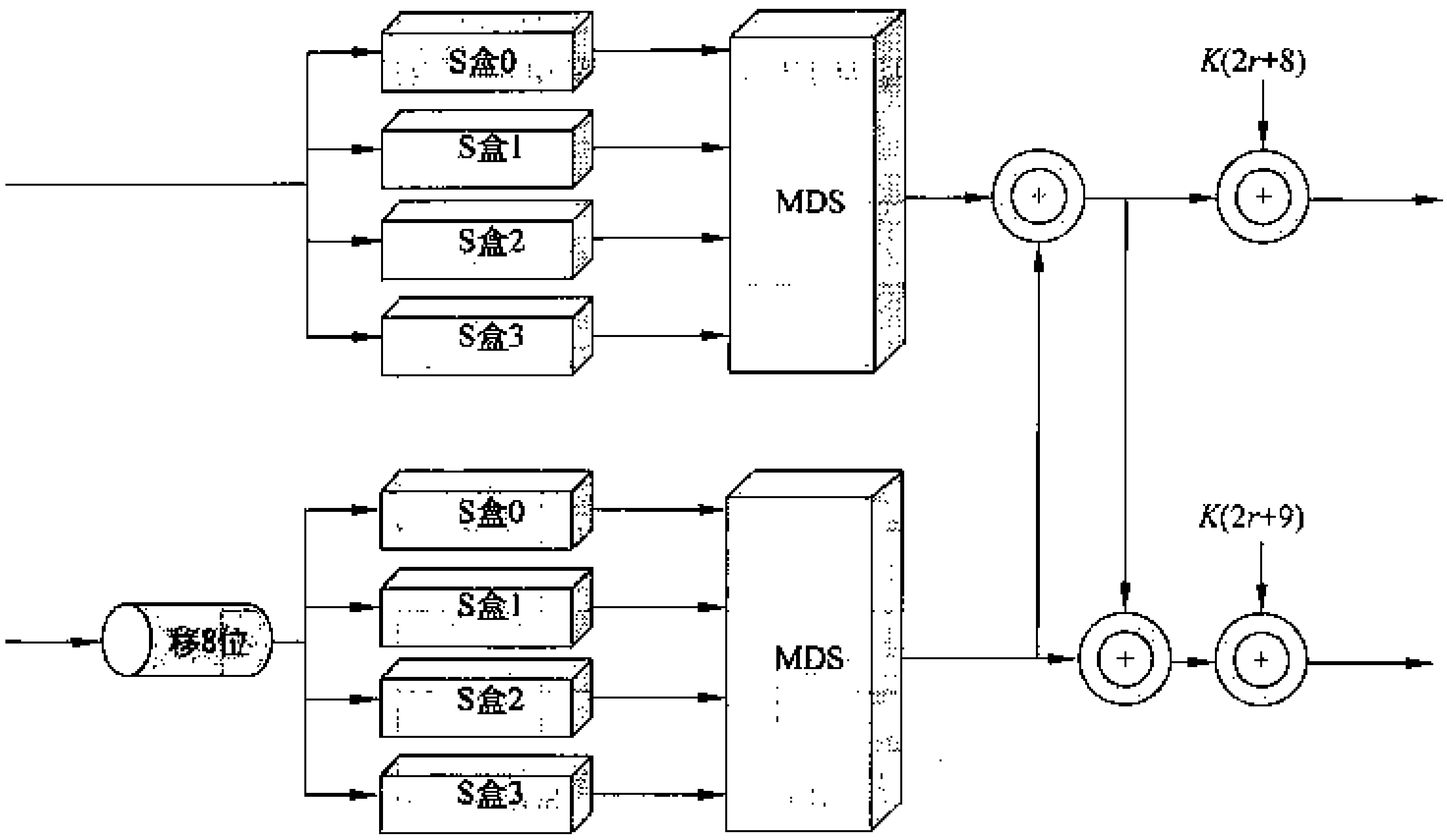


图 7.42 Twofish 的一个 F 盒

7.5.6.3 评估与性能

AES 的 5 个候选者公布后，它们的算法经历了广泛的测试。所有候选者都提供了较高的安全性，但这不是选择胜利者的惟一标准。加密速度以及硬件实现的特性也是重要的因素。如果某个算法的加密过程太慢，即使它能提供最高的安全性，用户也不会使用。

对于速度，NIST 为所有候选者提出了一个最低测试配置要求，以评估它们的性能。这个最低配置是 200MHz 的 Pentium Pro 系统，64MB 内存，运行在 Windows 95 环境下。要求算法用 Borland C++ 5.0 编写。这 5 个候选者的速度结果如图 7.43 所示。Rijndael 和 RC6 为最高。所有候选者的软件实现在 6 种不同的 Intel 系统、3 种 Alpha 系统、一种 HP 系统以及一种 Sun Sparc 系统上进行了测试。Rijndael 在其中的 5 种系统中排名第一，Twofish 在其中的 2 种系统中排名第一，RC6 在其余的 4 种系统中排名第一。显然，实现和

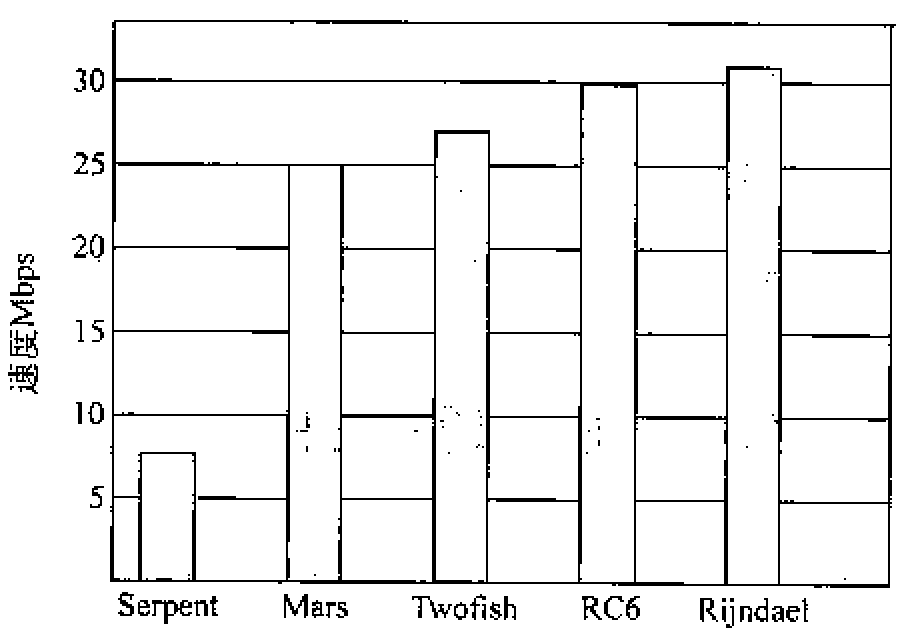


图 7.43 各个 AES 候选者的相对速度

体系结构在加密法的速度中起了很重要的作用。

速度不只是加密-解密的运行问题，它还是与子密钥的生成有关。块加密法都是使用了一种复杂的方法从初始密钥中生成子密钥的。各个候选者的子密钥生成测试结果如图 7.44 所示。结果并不令人奇怪，因为 RC6 具有最简单的子密钥生成算法，而 Twofish 的最为复杂。

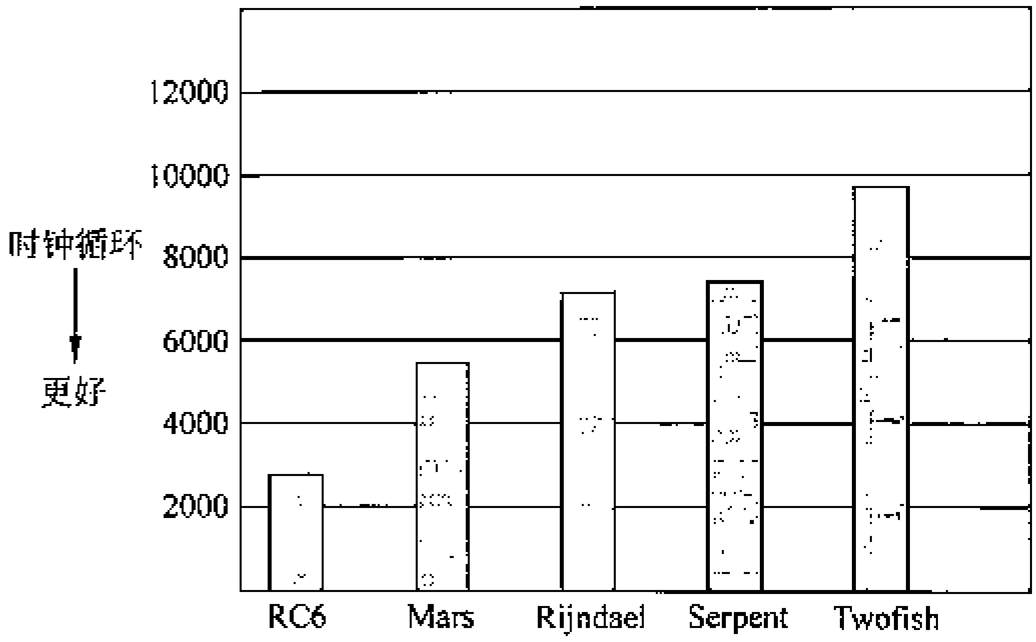


图 7.44　各个 AES 候选者的子密钥生成时间

由于 AES 要求用软件和硬件实现，因此其他的评估是关于硬件实现的大小和性能的。有关测试是在芯片和 FPGA(现场可编程门阵列)上进行的。要考虑的一个因素是需要芯片或 FPGA 的面积大小。图 7.45 显示了 George Mason 大学的团队在 Virtex FPGA 上实现各种算法时所需的面积大小。令人吃惊的是，Twofish 所需要的面积与 RC6 一样小。

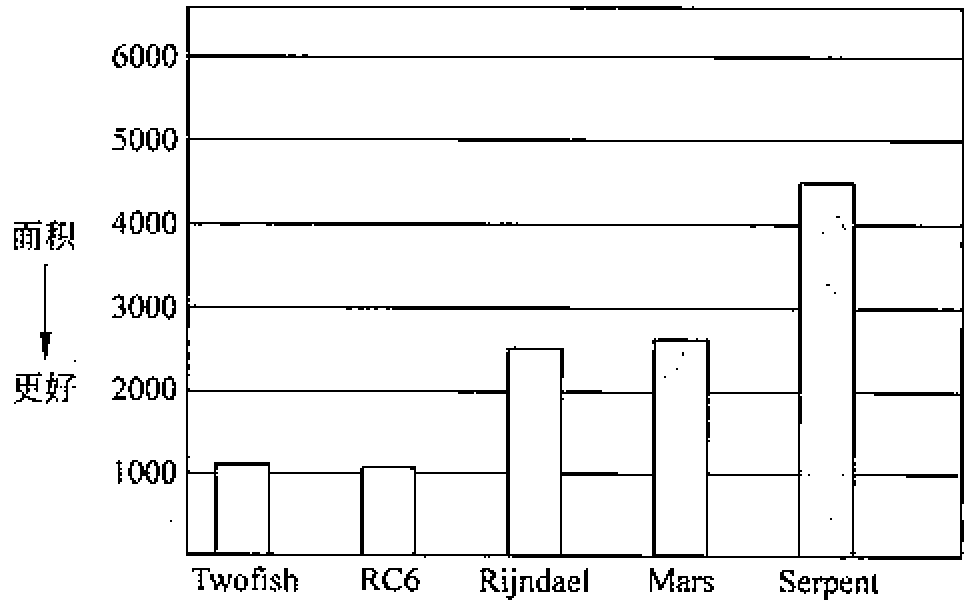


图 7.45　各个 AES 候选者所需的 FPGA 面积大小

7.6　块加密法的使用

至此，我们一直假设是 Bob 和 Alice，想安全地进行通信，选用一个常用的加密法和密钥，并用这些来保护我们的通信消息。这并没有什么错，而且有时也的确这么是做的。但

是，大多数计算机用户并不知道如何选用加密法或如何生成密钥，而在线通信中，每个人或多或少都有一些保密信息。他们要么对保护信息一无所知，要么没有时间或能力来选用自己的加密算法或密钥。于是，他们就使用一种特殊的网络，该网络是以一种透明但有效的方式进行保护信息的。

其中使用的一种加密法在第6章中已经进行了介绍。还记得WEP协议使用的是RC4流加密法吗？WEP协议对无线连接可能很好，但对有线网络呢？事实证明，要自动地保护网络通信，需要几个处理过程，其中之一就是IPSec。IPSec定义了一种方法，以实现在因特网上的安全通信。IPSec是一个相当详尽的处理过程。这里进行了简化，以便将重点放在其加密的使用上。在第9章中将介绍IPSec的一些认证过程。要了解其更全面的内容，请参考有关网络安全方面的教材。

7.6.1 网络连接概述

因特网是由成千上万的计算机以及其他相互连接在一起的设备组成的。例如，图7.46说明了路由器、服务器、公司网络和家庭计算机系统的使用情况。为了使不同的系统能协同工作，那么每个系统就得遵循一个通用的协议：每个系统必须能理解消息的格式，消息发送和接收的顺序，以及消息传输或接收时应采取的动作。

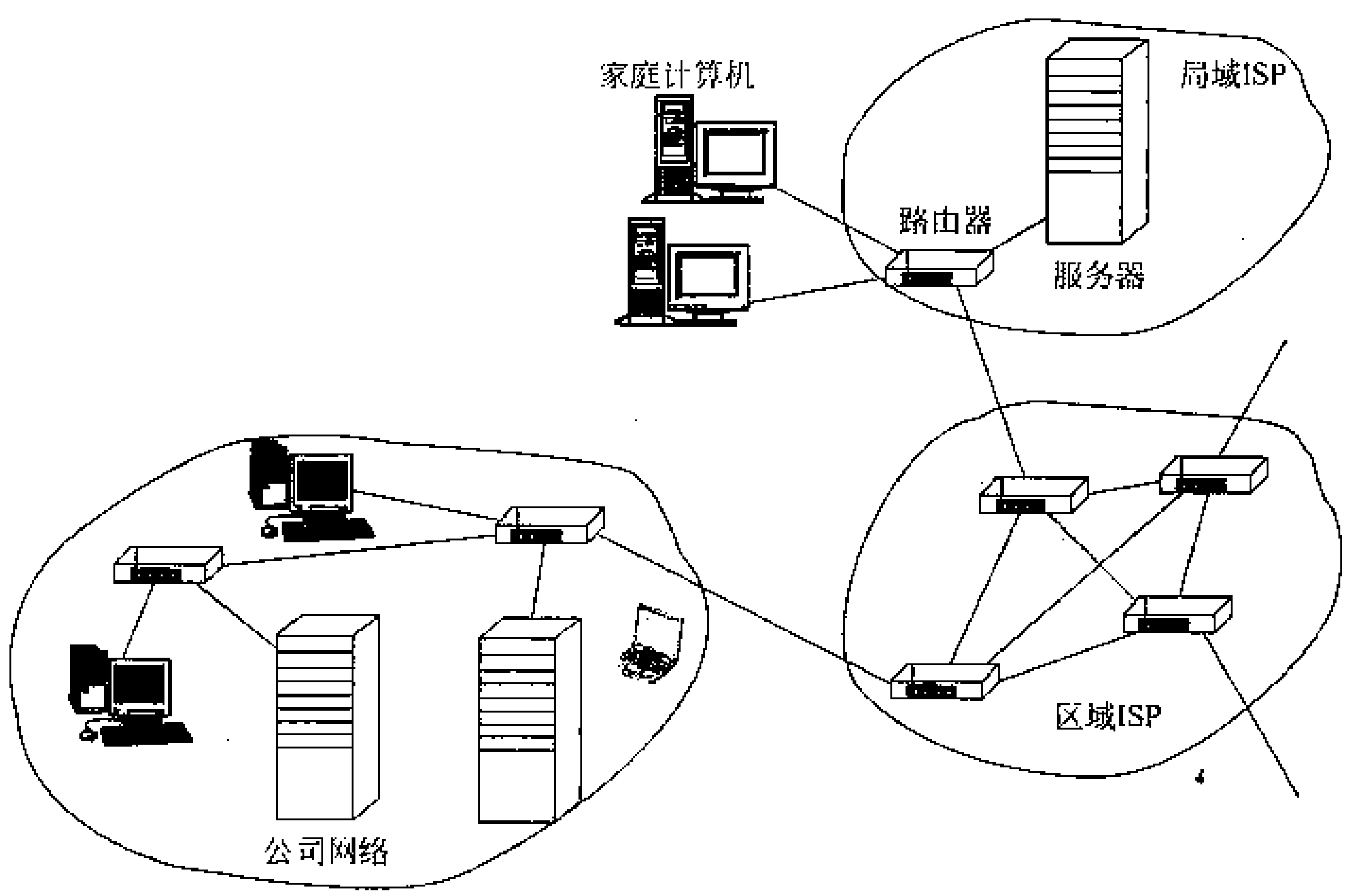


图7.46 一个小型的因特网示意图

要完成这个目标的方法很多，有关这方面的讨论需要涉及网络链路层的内容，而这与这里的主题相背离，我们这里的主题是理解块加密法的使用方法。因此，我们这里只讨论一个称为TCP的传输协议。

在因特网上，消息是以数据包发送的。一个数据包就是多个字节的集合，它包含了消息和寻址及控制消息。但是，当数据包在两台机器之间发送之前，必须先在两者之间建立一个

连接，这由 TCP 来完成。当某个主机要建立一个连接时，TCP 给目标机器发送一个特殊的数据包(称为请求消息)。该数据包的结构如图 7.47 所示。

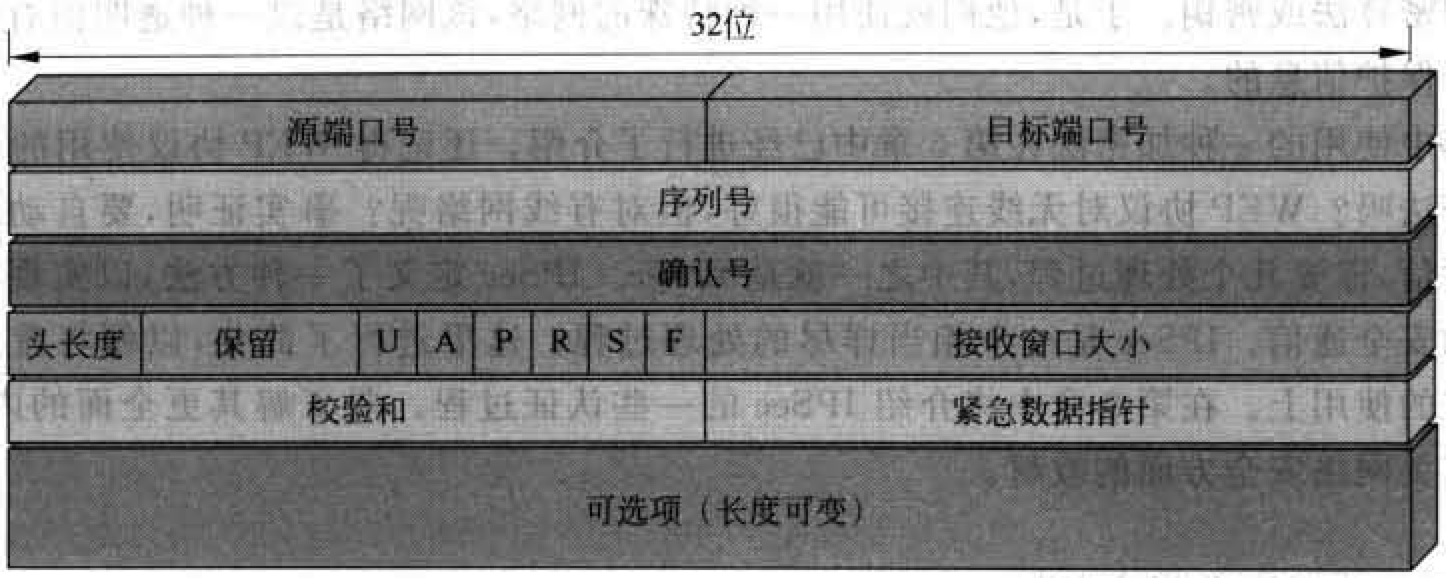


图 7.47 一个 TCP 数据包的头

TCP 头包含了一个端口号，它是一个与应用有关的数值。例如，文件传输应用(FTP)使用的是端口 21。当目标主机接收了连接请求后，返回一个包含了其惟一的套接字和端口号的消息，从而能确定两主机之间的虚拟连接。记住，这里只是一个简单的介绍，详细内容请参考有关资料。

7.6.2 IPSec

要使连接安全，可以采用 IPSec 协议，它给数据包添加了一些其他的信息。这些信息的作用是在两台机器之间创建一个**安全关联**(security association，SA)。一个 SA 记录了要求支持一个安全连接的所有信息。它包括验证与加密密钥、特定的加密法、IP 地址，等等。除 SA 外，IPSec 还使用两种基本的安全协议。这意味着它实际上往数据包中添加了另外两个头。第一个是**验证头**(authentication header，AH)，它使用一个哈希函数来实现数据包的验证(有关哈希函数的详细内容将在第 9 章介绍)。第二个是**封装安全负载**(encapsulating security payload，ESP)，它使用加密来维护数据包的完整性。ESP 使用块加密法(如三重 DES、AES、IDEA 等)，在传输之前将数据包加密，在接收时将它解密。因此，IPSec 创建一个 SA，使用带 AES 加密法的 ESP 来保证数据包的安全。图 7.48 显示了该协议将头信息添加到消息数据包中的情况。

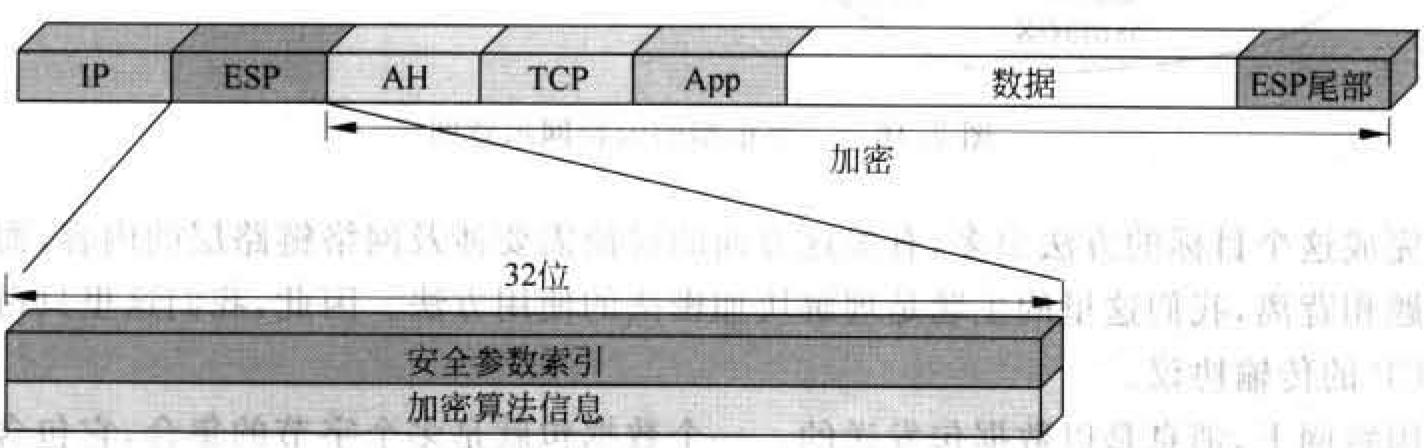


图 7.48 IPSec 中一个已加密的数据包

在图 7.48 中，同时还添加了其他几个头信息。App 头包含了与该数据相关的基本应用程序的信息。IP 头包含了源因特网地址和目标因特网地址、消息总长度等信息。AH 头包含了验证信息。ESP 头包含了加密算法及其必需参数的信息。ESP 尾部包含了填充位，以确保被加密段的大小与加密法所需的块大小相匹配。注意，除 IP 和 ESP 之外，所有其他数据都被加密。对 IP 加密是没有意义的，因为如果加密的话，两台机器之间的任何路由器都无法读取目标地址了。

所有这些中最重要的一点是，无须用户的干预。当 Bob 和 Alice 登录因特网并选择使用 IPSec 时，加密算法和密钥是由系统确定的。问题是：这是如何完成的？答案是，你得等到第 9 章才能见分晓。现在，我们知道了保护信息的这种方法比单纯选用一个加密法和密钥更安全。要使这些实现过程自动化，还有很多工作要做。

7.7 块加密法的历史简介

在加密法的悠久历史中，块加密法只占一小段。但是，在其短暂的历史上，块加密法出现了一些有趣又令人困惑的道德和伦理的两难境界。Clipper 芯片是最困难问题之一。

问题是在 1992 年因 AT&T 公司而起的。注意，AT&T 公司在密码学的历史中起了很重要的作用。当时 AT&T 公司宣布了 3600 TSD(Telephone Security Device，电话安全设备）——一种商用的用 DES 加密的电话附加设备——的产生。美国政府（尤其是 FBI 和美国国家安全局 NSA）表示对 TSD 的反对。由于合法的电话机是反犯罪和反恐的一个有力武器，美国政府担心 TSD 将削弱它的这种能力。NSA 采取的第一步是说服 AT&T 公司使用一种名为 Clipper 的硬件加密芯片，以换取政府购买的保证。NSA 买下整个 TSD 产品，并支付 AT&T 公司换用 Clipper 芯片的修缮费。这只是一个权宜之计。1993 年 5 月 15 日白宫宣布了整个处理方法。

事实证明，美国政府花了相当的时间来考虑设计与 AT&T 公司的 TSD 类似的替代品。为防不测，NSA 开发了一种保密的加密算法，称为 Skipjack，并将它嵌入在了 Clipper 芯片中。直到 1998 年人们才将 Skipjack 归类。它是一种类似于 DES 的块加密法，使用一个 80 位的密钥，作用于 64 位的块。它具有 32 轮的加密。由于最初它是保密的，在机密社区中引起了人们的很多关注。它具有一个只有 NSA 才知道的后门。自从它发布以来，人们也没有发现这个后门，但发现了它的一些弱点。但是，Skipjack 令人感兴趣的不是算法本身，而是建议的密钥管理过程。

美国政府希望能实现两件事：(1)通信设备制造商能自愿地采用 Clipper 芯片；(2)Clipper 芯片的密钥都保存在密钥托管(key-escrow)系统中。如果实现了这两点，那么美国政府就能确保有线电话机还是一个智能的信息收集工具。为了确保通信设备制造商采用 Clipper 芯片，美国政府要求政府购买的所有通信设备都嵌入 Clipper 芯片。美国政府希望的是，如此巨大的客户产生的市场购买力能促进 Clipper 芯片的使用。采用密钥托管系统——这是 Clipper 芯片处理的核心——则更为困难。

每个 Clipper 芯片含有一个惟一的标识码(UID)，一个 80 位的族密钥(family key，KF)，它对所有的 Clipper 芯片都通用，以及一个 80 位的**惟一芯片密钥**(unique chip key，

KU)。它还生成(这也是产生争论的地方)一个**法律强制访问字段**(law enforcement access field,LEAF),该字段与每个会话一起传输。LEAF 包含了还原会话密钥所需要的所有信息。创建 LEAF 的过程包括如下步骤:

1. 用芯片密钥 KU 加密会话密钥 KS;

2. 创建一个包含了 UID‖$\{KS\}_{KU}$‖的主密钥,哈希函数用族密钥 KF 加密,其中哈希为一个 16 位的认证者。

知道了 KU 和 KF 就可以用来还原**会话密钥**(KS,session key)。从这里密钥托管系统开始发挥作用了。制造商为每个 Clipper 芯片指定一个 UID 和 KU。KU 分成 KU_1 和 KU_2 两部分,这样 $KU=KU_1$ XOR KU_2。这两部分都用族密钥 KF 加密,并分给两个不同的代理。如果法律强制执行官员想进入某个通信,他们必须获得法院命令并出示给这两个代理。有了这个托管的密钥,他们就可以重构 KU。由于知道了 KF,就可以解开 LEAF 和 KS 了。

本节开始提到了道德和伦理的两难境界,这就是,为了有效地打击犯罪和恐怖活动,是否就允许政府访问加密的通信,还是公民有权无限制地使用功能强大的加密法?这是一个棘手的问题。赞成使用 Clipper 芯片的人认为,这种政策并不能有什么作用,因为仍然需要授权。因此,密钥托管是个人权利与政府保护其公民的职能之间的一种折衷方案。另一方面,很多人担心这将有损个人隐私的危险。业界担心将会失去功能强大的加密法的使用。Clipper 芯片技术的限制将削弱在有关产品国际市场销售的竞争力。

最后,调查显示,80%的美国人反对使用 Clipper 芯片。AT&T 公司关闭了这条产品线,美国国会也未能通过提交的法令。现在,有关的争论仍在继续。

7.8 本章小结

块加密法肯定比我们已经介绍过的经典加密法要复杂。它们甚至比现代的流加密法还要复杂。不仅是加密法本身更复杂,实现块加密法的过程——加密模式——也更复杂。Bob 和 Alice 对这些加密法印象很深。但是,他们应该记得从经典加密法中所吸取的教训——复杂并不能保证安全。毕竟,他们还记得,选用多种经典加密法都在 Eve 的努力下失败了。他们作出了一些糟糕的选择,Eve 可以破解他们的加密法,仍然可以阅读他们的消息。但是,正确使用块加密法,能够获得 Bob 和 Alice 一直想要的安全性。Bob 和 Alice 的失败突出说明了选择功能强的加密法并正确使用的必要性。

Eve 又一次明白了已知明文—密文攻击法的重要性。现代的块加密法功能太强了,只要正确使用,就不可能用强力攻击法来破解。密文攻击法也不像攻击经典加密法那样管用了。

总之,Bob、Alice 和 Eve 都懂得了加密法的强大功能与很多因素有关,其中很多是主观的。以下是一些这样的因素:

- 没有使用密钥,不能从密文得到明文。
- 没能出现明文攻击法比强力攻击法更好。
- 人们知道了算法也不应削弱加密法的强度。
- 加密算法必须包括在输入数据和密钥控制下的替换和置换操作。
- 明文中的冗余位组在密文中应完全打乱。

- 密文的长度应与明文的相同。
- 任何可能的密钥都应可生成一个强硬的密码。

最后，又有一种新的加密法类型和一些新的加密法添加到我们的加密法结构图中了，如图7.49所示。

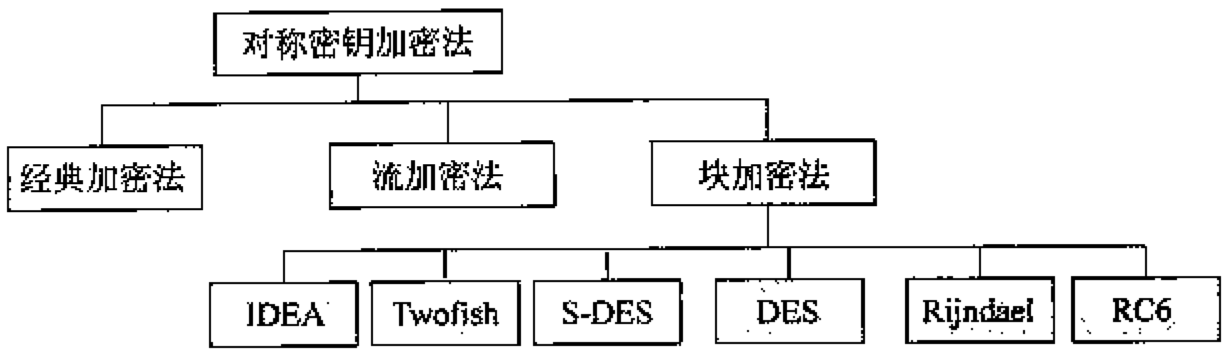


图7.49 本章介绍过的块加密法

7.9 重要术语

AES	高级加密标准——在经历长时间的检查和评估后，选择了 Rijndael。
ASIC	应用特定集成电路。
CBC	加密块链——像自动密钥加密法一样使用块加密法。
CTR	计数器模式——使用块加密法来生成类似于流加密法的密钥。
DES	数据加密标准。
差分密码分析法	一种用于破解块加密法的已知明文攻击法。
ECB	电子编码簿模式——用相同的密钥加密每个块。
FPGA	现场可编程门阵列。
ICM	整数计数器模式——像流加密法一样使用块加密法。
IV	初始向量——使用在一些块加密法中，用来增加安全性。
线性密码分析法	破解块加密法的已知明文攻击法。
中间点攻击法	中间点攻击法。
OCB	
OFM	
填充位	
乘积加密法	
S盒	替换盒，作为一种非线性转换用在块加密法中。
TCP	传输控制协议。

习　题

1. 全为1的DES密钥的子密钥是什么？全为0的呢？1和0各一半的呢？
2. S-DES有一些脆弱的密钥。请至少找出一个来(0除外)。

3. 请破解以下的 S-DES 消息：

11100011001101011110000011000 1
0100101110001010100011110010 11
00001110000011010001001010101 1
0100011101101100101110110100 01
0010101011010001001111000000 01
01000000100000001000101010111 0
00001101000100101010111000000 0
1111100011110000101110010111 10
00101110110100010011110011100 0
000011111001000000001010100110
01111110101101100111

4. Bob 和 Alice 尝试使用双重 S-DES，认为 Eve 破解它是破解单个 S-DES 的难度的两倍。请破解以下消息：

1010010010001101100100111101
00000001011011111001000 11111
1000010111010000110101001011
0101110001101000010100011111
1000110110111110110100000001
1111100001011101000010000101
0001011011000110100011011011
1110000100011011010111000110
1101000010010001111111010001
11110010011110110101

5. Bob 和 Alice 想对 S-DES 加密法进行评估。他们首先查看密钥中有多少个位发生了变化。他们发现，对每个小写字符应用密钥 28、76、84、88、92、93、94、124，在所生成的密文，发生了变化的位的平均数各不相同。也就是说，他们运行这样一种测试：明文字符为"a"，分别尝试这 8 个密钥。然后对字符"b"进行同样的测试，等等。

(a) 密文中平均有多少位变化了？

(b) 为什么选用这 8 个密钥？

(c) 对 S-DES 的性能得出了什么结论？

6. Bob 和 Alice 决定对 S-DES 加密法进行类似的评估。他们首先对明文"This is a test"严格的密钥雪崩条件测试，使用的密钥为：abcdefgh、qbcdefgh 和 arcdefgh。他们发现了什么，得出了什么结论？

7. 最后，Bob 和 Alice 测试严格的明文雪崩条件。他们使用密钥"abcdefgh"，明文样本为"This"、"Dhis"和"Txis"。他们发现了什么，得出了什么结论？

8. 请指出 DES 的 S 盒 1 的不同分布。

9. Bob 和 Alice 决定测试 S-DES 输出的随机性。他们输入了如下明文：

This is a test of s-des randomness. The ciphertext should appear to be a random set of

binary bits if the cipher is to be trusted.

他们使用密钥"1011"来运行S-DES，然后应用CAP软件对密文进行测试。他们发现了什么，得出了什么结论？

10. Bob和Alice使用相同的明文在DES上重复随机性测试。他们使用的密钥是"retestit"。他们发现了什么，得出了什么结论？

11. CAP软件实现了输出反馈模式的S-DES，即是作为一个密钥流生成器。使用密钥"1103"和初始输入"z"，生成1000位。运行该随机性测试并报告你的结论。

12. 使用DES在输出-反馈模式(CAP软件实际上已实现了)下重复上面问题11，使用的DES密钥为"abcedfgh"，初始数据为"abcedfgh"。

13. 请试比较使用ECB与使用CBC对密文中发生单个位错误的影响。

14. 对DES的S盒2，输入为100100，请指出有多少个输出位与初始输入位不同，也就是说，100100与000100之间有多少个输出位不同？

15. 在一个图中，为S盒2的4个行标出行号和S盒的值。请给出你观察的结果评价。

16. 如果输入为"Alice just do it"，第一个阶段使用的密钥01 01 01 01 20 20 20 20 03 03 03 03 60 60 60 60，那么第一个阶段的AES输出是什么？

17. 假设AES的密钥为30014fd1 69e31044 1782e4b1 23aa4018，第一轮的密钥是什么？

18. 破解AES。请与你的教师和本书作者分享你的评价。

第 8 章　公钥加密法

8.0　概　　述

在已经介绍过的所有经典和现代加密法中，一个主要的问题是密钥。它们都只有一个密钥，这个密钥既用来加密，也用来解密。这看上去很实用也很方便。但问题是，每个有权访问明文的人都必须具有该密钥。密钥的发布成了这些加密法的一个弱点。因为如果一个粗心的用户泄漏了密钥，那么就等于泄漏了所有密文。这个问题在一种新型的加密法中得到了解决，那就是公钥加密算法。该算法是由 Diffie、Hellmann 于 1976 年提出的。这种加密法有两个不同的密钥：一个用来加密，另一个用来解密。加密密钥可以是公开的，每个人都可以使用它来加密，只有解密密钥是保密的。这也称为不对称密钥加密法。

为了形象地描述这个问题，让我们假设 Bob 是处在这样一个虚拟世界中：所有电话机都是成对连接的，只用一个按钮来保护个人隐私。这样就不需要标准号码簿了，因为每部电话机都只与另外一部相连。这意味着，要与 Alice 通话，Bob 就必须与她共享一对电话机。如果 Bob 想与 Paul 通话，他们就得共享另一对电话机。最后，可怜的 Bob 就得携带大量的电话机了，如图 8.1 所示。当然，每条电话连接都很安全，除非某部电话机被 Eve 拾到了。如果 Bob 必须为每个要发送的消息都用不同的密钥，这就相当于他与每个人通话必须有一部不同的电话机一样。也就是说，他就得有与 Alice 通信的密钥，有与 Paul 通信的密钥，等等。最后，他要么忘记了这些密钥，要么写在某个可能被 Eve 发现的地方。管理密钥成了一个严峻而困难的问题，就像是不得不携带大量的电话机一样。

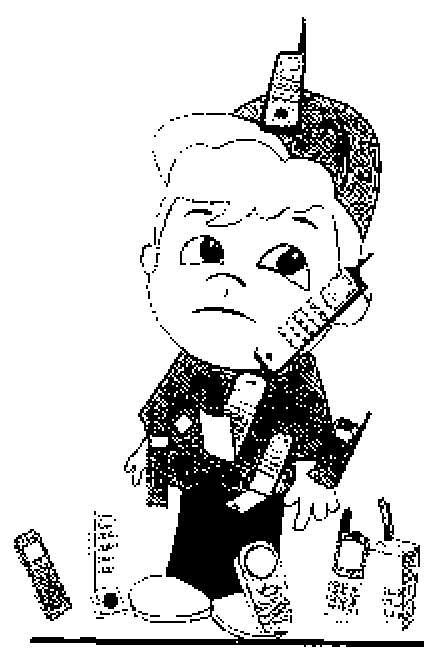


图 8.1　可怜的 Bob

当然，Bob 并不需要携带多部电话机，因为在现实世界中，他所需要的是一部可以拨打多个号码的电话机，以及一本记录着电话号码的电话簿。现实世界中的 Bob 如图 8.2 所示。瞧，他多轻松，他只需一部电话机即可。这在很多方面与公钥的概念相似。加密密钥保存在公开场所，这样任何人都可以查找所需的密钥。本章将介绍这种新型加密法的一些特征，以及通常是如何使用的。

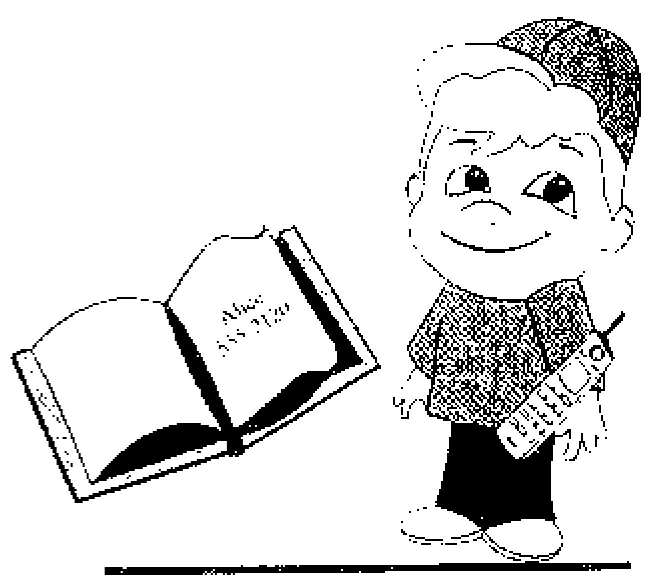


图 8.2　现实中的 Bob

8.1　公钥的加密解密过程

Alice 和 Bob 希望使用公钥加密法是因为它有多个密钥。他们喜欢这种新系统在于，他们每人都将拥有一个保密密钥和一个公钥。Bob 将公钥给 Alice，与 Alice 共享该密钥。他们甚至将该密钥写在“公钥簿”上。Eve 也可以获得该公钥。这既不会泄漏 Bob 的内容，也不会泄漏 Alice 的内容。公钥（PK）系统的优点是，一旦共享了公钥，Alice 就可以发送消息给 Bob，该消息使用 Bob 的公钥进行了加密。只有 Bob 才能解密该消息，因为只有 Bob 才有相应的私钥。公钥的加密解密过程如图 8.3 所示。

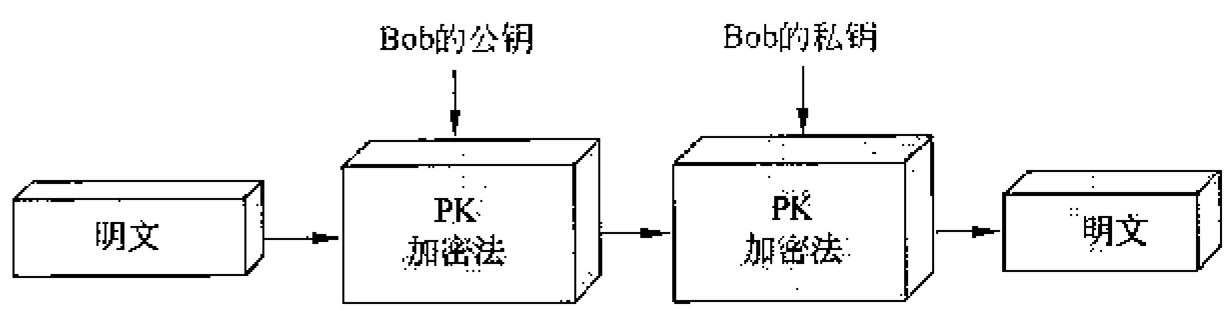


图 8.3　公钥的加密解密过程

一个 PK 系统的要求如下：

- 它应很容易生成公钥和私钥；
- 它应很容易加密和解密；
- 从公钥很难得出私钥；
- 利用密钥和公钥很难得出明文。

公钥这一概念的真正贡献是，它减少了网络用户必须管理的密钥数。使用如 AES 之类的对称加密系统，每对通信链接要求一个会话密钥。图 8.4 是一个包含有 5 个用户的网络，如果使用对称加密法进行通信，总共需要 10 个保密密钥，每个用户都得记住（并保密）4 个密钥。但是，在公钥系统中，每个用户只需记住他们自己的私钥，从公共场所查找所需的公钥即可。公钥的另一个优点体现在有一个新用户加密网络组中的时候。如果第 6 个用户要加入到如图 8.4 所示的网络中，就需要 5 个新的密钥，如图 8.5 所示。但是，在公钥系统中，

只需要在记录公钥的地方添加新用户的公钥即可。公钥系统解决的另一个问题是密钥的创建。在对称密钥系统中，每个用户都得协商一个通用保密密钥值。如果他们不妥善保管，这就可能成为整个加密系统的一个弱点。相反，在公钥系统中，每个用户既创建自己的密钥，也创建公钥，用户之间无需因此进行通信。另外，公钥系统还有其他用途，如数字签名，这些将在第 9 章中介绍。

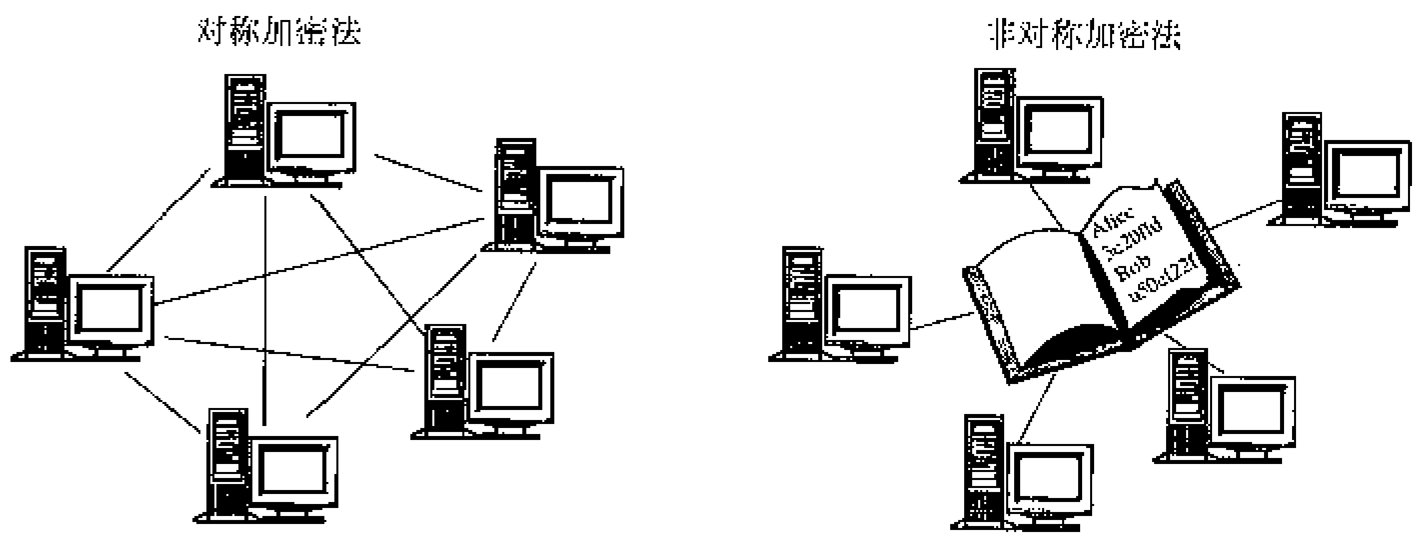


图 8.4 密钥需求的情况

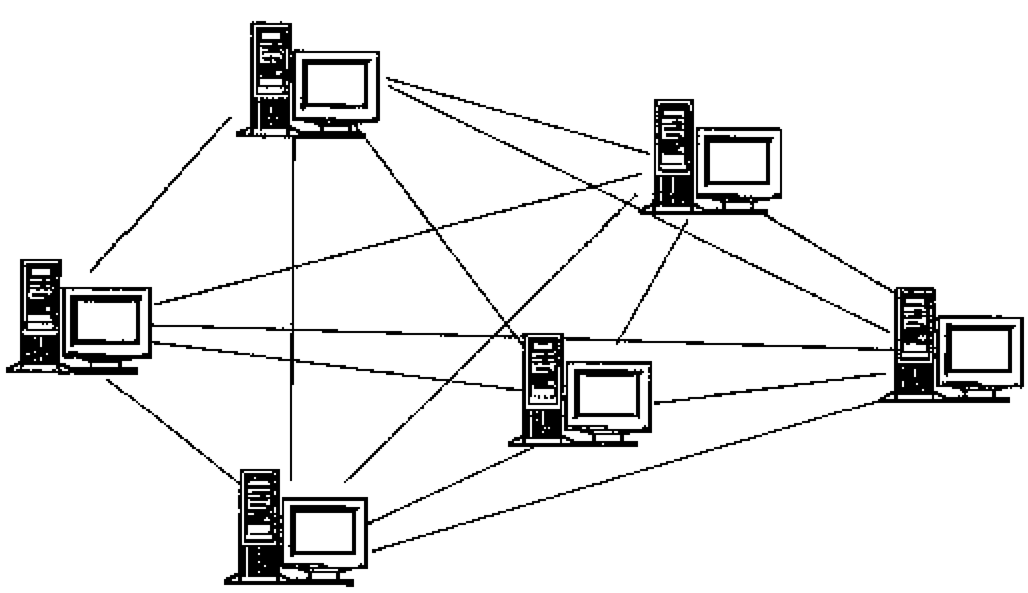


图 8.5 添加一个用户的情况

8.2 RSA

实现公钥有多种方法和算法。大多数都是基于求解难题的。也就是说，是很难解决的问题。人们往往把大数字的因子分解或找出一个数的对数之类的问题作为公钥系统的基础。但是，要谨记的是，有时候并不能证明这些问题就真的是不可解的。这些问题只是看上去是不可解的，因为经历了多年后仍未能找到一个简单的解决办法。一旦找到了一个解决办法，那么基于这个问题的加密算法就不再安全或有用了。

最常见的公钥加密法之一是 RSA（以其发明者 Rivest、Shamir 和 Adleman 命名的）。它是基于指数加密概念的。指数加密就是使用乘法来生成密钥。其过程是首先将明文字符转换成数字，即将明文字符的 ASCII 二进制表示转换成相等的整数。计算除明文整数值的 e 次幂，再对 n 取模，即可计算出密文：

$$C = P^e \bmod n$$

e 和 n 的值就是加密法的密钥。当然，如果要使之对加密法有用，这个过程必须是可逆的。指数加密法可以用另一个密钥 d 来解密，如下所示：

$$P = C^d \bmod n$$

d 的值可以通过求解下面等式得出：

$$ed \bmod \varphi(n) = 1$$

其中，φ 是欧拉函数。它定义为整数 j 的数目，其中 j 小于或等于 n，且 $\mathrm{GCD}(j,n)=1$。也就是说，整数个数小于 n，且与 n 没有公因子。如果 n 是两个素数 p 和 q 的乘积（即 $n=pq$），那么 $\varphi(n)=(p-1)(q-1)$。求解 $\varphi(n)$ 的数学方法是欧拉定理。该定理表明，如果 $\mathrm{GCD}(x,n)=1$，那么：

$$x^{\varphi(n)} \bmod n = 1$$

正如后面将介绍的那样，该定理也是解密 RSA 的方法基础。

Rivest、Shamir 和 Adleman 采用了这个基本模型，并把它应用于正式的公钥加密法中，定义了一个特殊的方法来确定 e、n 和 d 的值。n 的值通过选取两个大素数 p 和 q，且 $n=pq$ 来确定的。由于 n 是公钥的一部分，因此要将它公开。但 p 和 q 必须保密。然后在 $\min(p,q)$ 和 $(p-1)(q-1)$ 之间选取另一个数 d，该数与 $(p-1)(q-1)$ 互质（即两者之间没有公因子）。用如下等式可以计算出 e：

$$ed \bmod (p-1)(q-1) = 1$$

将 e 作为公钥的一部分而公开，但 d 得保密。

以上就是确定公开和私钥的过程。RSA 的加密过程就像是一个指数加密。把明文分成若干个消息块，这样每个块就可用一个大的数字表示。消息块 M 的密文 C 可用如下等式算出：

$$C = M^e \bmod n$$

由于 e 和 n 是公钥，任何人都可以生成密文。

RSA 的解密过程是利用私钥 d，通过如下等式来求解出 M 的：

$$M = C^d \bmod n$$

这是成立的，因为：

$$C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

由于 $ed \bmod \varphi(n)=1$，这说明 ed 是 1 加上 $\varphi(n)$ 的若干倍之和，因此必定存在这样的整数 r，使得：

$$ed = 1 + r\varphi(n)$$

这意味着：

$$M^{ed} \bmod n = M^{1+r\varphi(n)} \bmod n = M^1(M^{\varphi(n)})^r \bmod n = M(1)^r \bmod n$$

综合以上这些，可以得出：

$$C^d \bmod n = M$$

由于 d 是私用的，因此只有 d 的拥有者才能解开密文。这个过程是安全的，因为，要找出 d，就必须知道 p 和 q。找出 p 和 q 的惟一方法是将 n 分解因子。如果 n 很大，那么分解就很困难（正如后面你将看到的那样）。

Bob 和 Alice 想使用 RSA 加密法，因此他们先拿一个简单的样本范例进行尝试，以便

了解该加密法是如何工作的。Bob 选取了两个小素数 11 和 23。这两个素数的乘积确定了 n：

$$n = 11 \times 23 = 253$$

接下来就是选取私钥 d，d 与$(p-1)(q-1)$（也就是 $10 \times 22 = 220$）互质。这样的数有很多个，但 Bob 选取了 19。另一个公钥 e 可用如下等式求出：

$$19e \bmod 220 = 1$$

答案是 $e=139$。Bob 只是告诉 Alice，e 和 n 的值为(139，253)。Alice 决定来检测一下该加密法，给 Bob 发送消息"Hi"。在 ASCII 中，该消息用 16 位二进制表示为 01001000 01011001，将它转换成十进制数为 72 105。然后，Alice 使用 Bob 的公钥来转换这两个数：

$$72^{139} \bmod 253 = 2$$

$$105^{139} \bmod 253 = 101$$

Bob 介绍到该消息为(2;101)，然后用私钥还原成原来的文字：

$$2^{19} \bmod 253 = 72$$

$$101^{19} \bmod 253 = 105$$

将这两个数转换成二进制，并在 ASCII 表中查找相应的字符，可以知道发送给 Bob 的真正消息是"Hi"。

Bob 和 Alice 发现 RSA 能很好地工作，但是，他们不可能真的使用这么小的数字，因为这样 Eve 将能很容易从公钥发现私钥。例如，如果 Eve 知道了 Bob 的公钥为(139，253)，那么她就可以尝试分解 253 以便得出 p 和 q 的值。最简单的方法是用小于 16 的每个素数去除 253(因为 16 是大于 253 的平方根的最小整数)。用不了多少时间，Eve 就发现 11 除 253 得 23。知道了 p 和 q 的值，也就知道了$(p-1)(q-1)=220$。由于已经知道了 e 的值为 139，Eve 要做的就是求解以下等式以得出 d(私钥)的值：

$$139d \bmod 220 = 1$$

于是很容易得出 d 为 19。这意味着 Bob 和 Alice 如果真的要使用 RSA，他们就必须选择两个很大的素数，从而使 n 几乎不可分解因子。

例如，Bob 可能选择：

$$p = 3490529510847650949147849619903898133417764638493387843990820577$$

$$q = 32769132993266709549961988190834461413177642967992942539798288533$$

这样 n 就是一个很大的数了：

$$n = 114381625757888867669235779976146612010218296721242362562561842935706935245733897830597123563958705058989075147599290026879543541$$

事实上，这也是 p 和 q 的一个糟糕选择，因为这个 n 值是 RSA 公司作为 RSA-129 而公开的，该数于 1994 年被成功分解了。在实际应用中，如果要想不让 Eve 发现他们的密钥，Bob 和 Alice 应选择多于 100 个数字的数。

因此，在实现 RSA 之前，Bob 和 Alice 面临一个关键的问题，那就是，他们需要找两个大素数，利用等式 $ed = 1 \bmod (p-1)(q-1)$ 求解出有关的值。由于计算器和一般的计算机程序无法计算这么大的整数，因此他们需要一些能执行长整数数字运算的方法。要解决这个问题，他们需要掌握一些数字理论方面的知识。下面一节就来介绍一下这方面的内容。

8.3 数字理论概述

为了更好地学习数字理论方面的知识，有必要以实际的大整数（含有 100 多个数字）为示例。这要求有特殊的软件。对于 Bob 和 Alice 来说，幸运的是，CAP 软件正好满足了他们的需要。从 CAP 的主菜单中选取 Integer 菜单项，打开 Long Integer Routines 窗口，如图 8.6 所示。该软件只是给了他们一个开端，要使用 RSA，需要计算反模运算和寻找大素数。

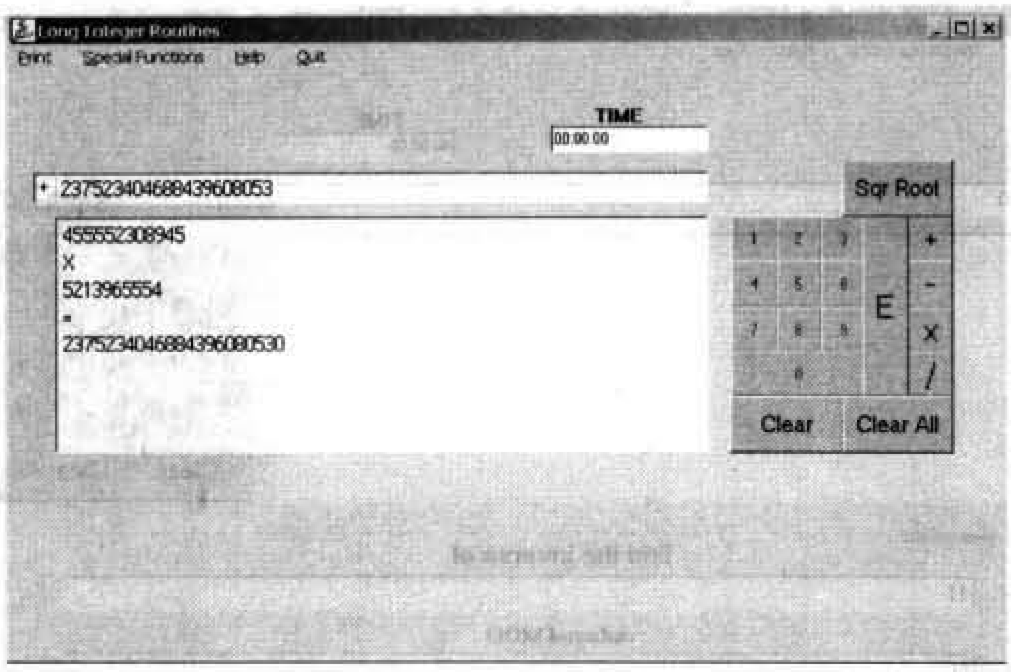


图 8.6 Long Integer Routines 窗口

8.3.1 反模运算

要使用 RSA，需要选取两个大素数（p 和 q）和公钥（e）。私钥 d 是公钥的反模运算，通过求解 $ed = 1 \bmod (p-1)(q-1)$ 得出。

这可以使用数字理论中的欧几里得算法来求解。欧几里得算法用于计算两个正整数的最大公约数。该算法的一种扩展形式是用来确定一个数的反模计算。假设 $ed \bmod m=1$，用以下步骤可计算出 d 的值：

1. $(x_1, x_2, x_3) \leftarrow (1, 0, m)$ 和 $(y_1, y_2, y_3) \leftarrow (0, 1, e)$。
2. 如果 $y_3=0$，那么这说明不可逆，停止；如果 $y_3=1$，那么 $d=y_2$，停止；否则，继续。
3. 设 $Q=x_3/y_3$（整数除）。
4. $(t_1, t_2, t_3) \leftarrow (x_1-Qy_1, x_2-Qy_2, x_3-Qy_3)$。
5. $(x_1, x_2, x_3) \leftarrow (y_1, y_2, y_3)$。
6. $(y_1, y_2, y_3) \leftarrow (t_1, t_2, t_3)$。
7. 返回第 2 步。

图 8.7 说明了求解 $11d \bmod 51=1$ 的步骤。

Q	X_1	X_2	X_3	Y_1	Y_2	Y_3
—	1	0	51	0	1	11
4	0	1	11	1	−4	7
1	1	−4	7	−1	5	4
1	−1	5	4	2	−9	3
1	2	−9	3	−3	14	1

图 8.7 扩展的欧几里得算法

如果是较小的整数，手工就可以很容易实现该算法，但对于大整数，则需要使用计算机了。CAP 软件提供了必要的工具，如图 8.8 所示。

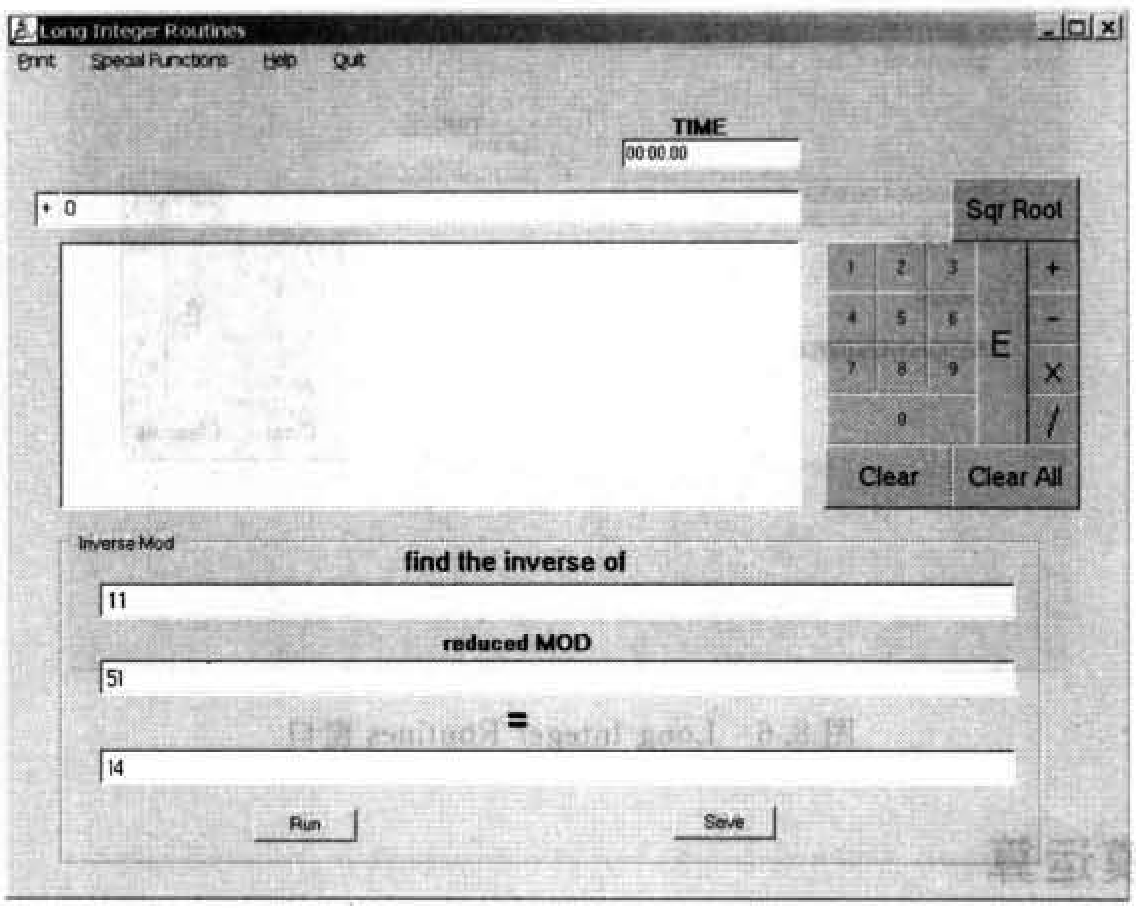


图 8.8 利用 CAP 软件实现反模运算

8.3.2 素数问题

Bob 和 Alice 面临的下一个问题是找到可作为公钥和私钥的两个大素数。从素数表中可以很容易地查找小素数，以作为 RSA 的密钥，但从安全的角度来说，这是不可取的。他们需要找大素数，并证明它们的确是素数，以便生成安全的 RSA 密钥。例如，他们可能尝试使用 12553(这个数太小了，但这里只是作为一个示例而已)。它是素数吗？要证明它是否是素数的一种方法是，用小于其平方根的每个数去除 12553。只要有一个数能整除 12553，那么 12553 就不是素数，否则，它就是素数。12553 的平方根是 112.04，因此可用 2～113 的每个数去除 12553。结果证明 12553 的确是一个素数，可以用来生成 RSA 密钥。Bob 和 Alice 决定尝试一个更大的数 13 952 598 148 481。手工计算将花费很长的时间。但是，CAP 软件提供了尾数除法测试，它是 Long Integer Routines 窗口中 Special Functions 的一个选项。图 8.9 可用来验证某个数是否为素数。显然，用尾数除法来验证 RSA 所需的大整数是不可取的。

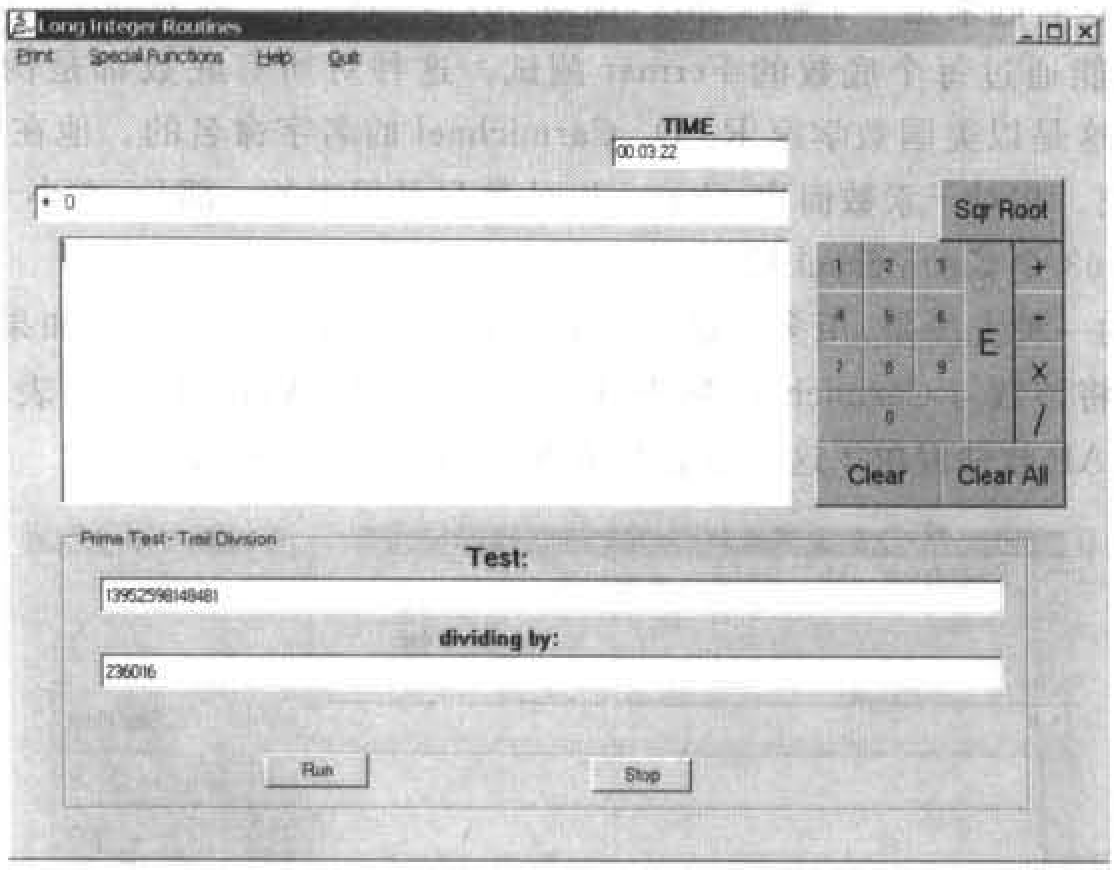


图 8.9 Long Integer Routines 窗口

这里介绍一个可稍微加速尾数除法的技巧，就是不必用从 2 到该数的平方根之间的每个数去除该数，而只需用该范围内的每个素数去除该数即可。由于可能并不知道所有的素数，这里介绍一个简单的公式，用来生成所有素数和一些合数，从而减少用于尾数除法的总个数。这个公式的说明如下：

- 除以 2、3 和 5；
- 对所有整数 k，除以 $6k-1$ 和 $6k+1$。

CAP 软件也提供了这种测试方法，也的确加快了计算速度，但这还是不能满足 Bob 和 Alice 对素数的要求。

如果 Bob 和 Alice 要使用 RSA，需要一个更快、更有效的素数测试方法。人们已经开发了多种素数测试方法，这些方法都是基于对素数的特性分析的。17 世纪法国数学家 Pierre Fermat 提出了一种方法。他指出，给定两个数，即底数 a 和素数 p，其中 a 不能被 p 整除，那么应有 $a^{p-1} \bmod p=1$。例如，以 6 为底数，7 为素数(6 不能被 7 整除)，那么：

$$6^{7-1} \bmod 7 = 6^6 \bmod 7 = 46656 \bmod 7 = 1$$

这就是 Fermat 的 Little 定理。它提出了一种验证素数的方法。如果某个数 c 可能是素数，那么选取一个底数(假设为 2)，验算 $2^{c-1} \bmod c=1$ 是否成立。Bob 和 Alice 认为这是一个好办法，因此他们用 341 来进行尝试。结果 $2^{340} \bmod 341=1$，他们认为 341 是素数，直到有人告诉他们 $11\times31=341$。因此 341 不是素数，这个简单的测试方法失败了。这里的问题不是出在 Fermat 的定理上，而是使用方法上。该定理并不是说如果 $a^{p-1} \bmod p=1$，那么 p 就肯定是素数。但是，尽管该定理对 341 无效，但对所有素数是有效的(译注：所有素数都能使该等式成立，但能使该等式成立的并不一定是素数)。人们把像 341 这样的合数称为伪素数(pseudoprime)。例如 91 也是一个以底数为 3 的伪素数，因为 $3^{90} \bmod 91=1$。

如果用两个不同的底数进行两次测试，就可以避免伪素数。也就是说，对底数 2 是伪素数，对底数 3 则不是。由于 341 是底数 2 的伪素数，因此再尝试底数 3，可以得出 3^{340} mod

341 不等于 1。这看起来是一个解决办法，但对 561 又不行了。因为 $561=3\times11\times17$，不是素数，但它的确能通过每个底数的 Fermat 测试。这种对所有底数都是伪素数的数称为 Carmichael 数，这是以美国数学家 R. D. Carmichael 的名字命名的。他在 1909 年对这些素数进行了研究。相对于素数而言，Carmichael 数只是很少的一部分，在小于 25×10^9 的所有数中，只有 2163 个 Carmichael 数。

检测素数的一种方法是，用多个底数对目标数进行 Fermat 测试。如果通过了所有底数测试，然后再将该数与 Carmichael 数表进行比较。如果该数不位于该表中，那么它肯定是一个素数。CAP 软件提供了这种素数测试方法，如图 8.10 所示。

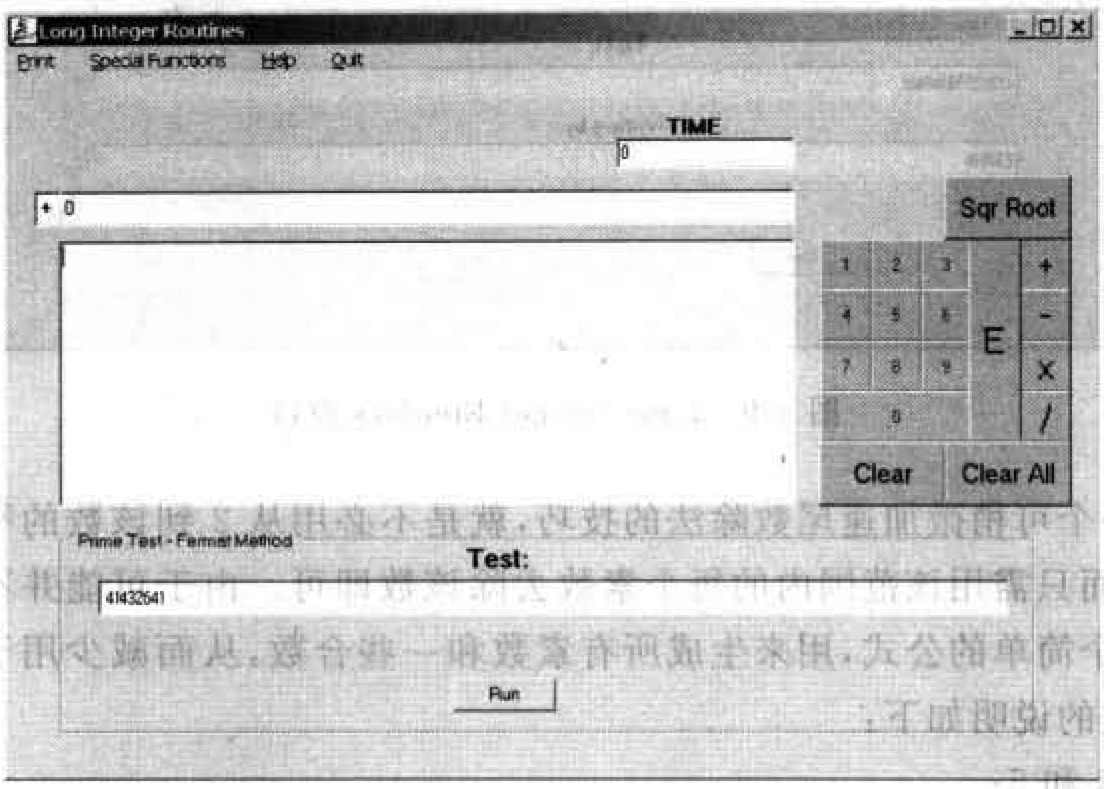


图 8.10　CAP 软件的 Fermat 素数测试法

这种方法存在的问题是，能测试的数受到了 Carmichael 数表的限制。如果要快速地确认任意大小的素数，就需要更通用的方法。这种方法分别由 Lehmann 于 1982 年和 Peralta 于 1985 年发现。这种测试方法是基于概率之上的。也就是说，如果某个数 p 通过了测试，那么 p 不是素数的概率就相当小了。所使用的方法是在 $1\sim p-1$ 中选 100 个随机数，如果：

(1) $a_i^{(p-1)/2}=1$，其中 $i=1\sim100$；

(2) 或者，$a_i^{(p-1)/2} <> 1$ 或 -1，其中 $i=1\sim100$。

那么 p 可能是合数。

相反，如果：

$$a_i^{(p-1)/2}=1 \text{ 或 } -1,\quad \text{其中 } i=1\sim100(\text{对于某些 } i, \text{等于 } -1)$$

那么 p 可能是素数。

如果能分解出一些小的素数因子，那么上面的方法就能很好工作了。CAP 软件同样实现了这种方法。

使用最广的方法是 Miller-Rabin 测试法。它是类似于 Lehmann 测试法的另一种素数测试法，也是基于概率的。它是基于以下知识的：给定一个奇素数 n，其中 $n-1=2^s r$（r 为奇数），以及一个随机整数 a，且 $\mathrm{GCD}(a,n)=1$（即 a 和 n 没有公因子），那么，要么

$$a^r \bmod n = 1$$

要么

$$a^{2^j r} \bmod n = 1 \quad (\text{其中}, j = 0 \sim s-1)$$

因此，对于一个可能是素数的数 n，找出 s 和 r，选取一个值 a，如果 $a^r \bmod n=1$ 成立，那么 n 为素数的概率为 0.25。如果上面两式都不成立，那么 n 就是合数。如果 n 通过了 m 个不同的 a 值，那么 n 为素数的概率为 $(0.25)^m$。CAP 软件实现了 Miller-Rabin 测试法，它尝试的是 100 个 a 值。

Bob 和 Alice 真正面临的问题不只是要测试某个数是否为素数，还有找出一对足够大的素数来构建 RSA 密钥。他们并不想自己去猜测数字，然后再去检测它们。因此他们选用了一种查找素数的常见方法。他们使用一个程序(如 CAP 软件)，该程序可生成 100 个大素数，当确定了某个为素数时，程序停止运行。图 8.11 显示了一个 15 位的素数，它是用 CAP 软件基于 Lehmann 测试方法得到的。注意，它花费了不少的时间，得到的却是一个相对较小的素数。CAP 软件还提供了一个类似的程序，它是基于 Miller-Rabin 测试方法的。

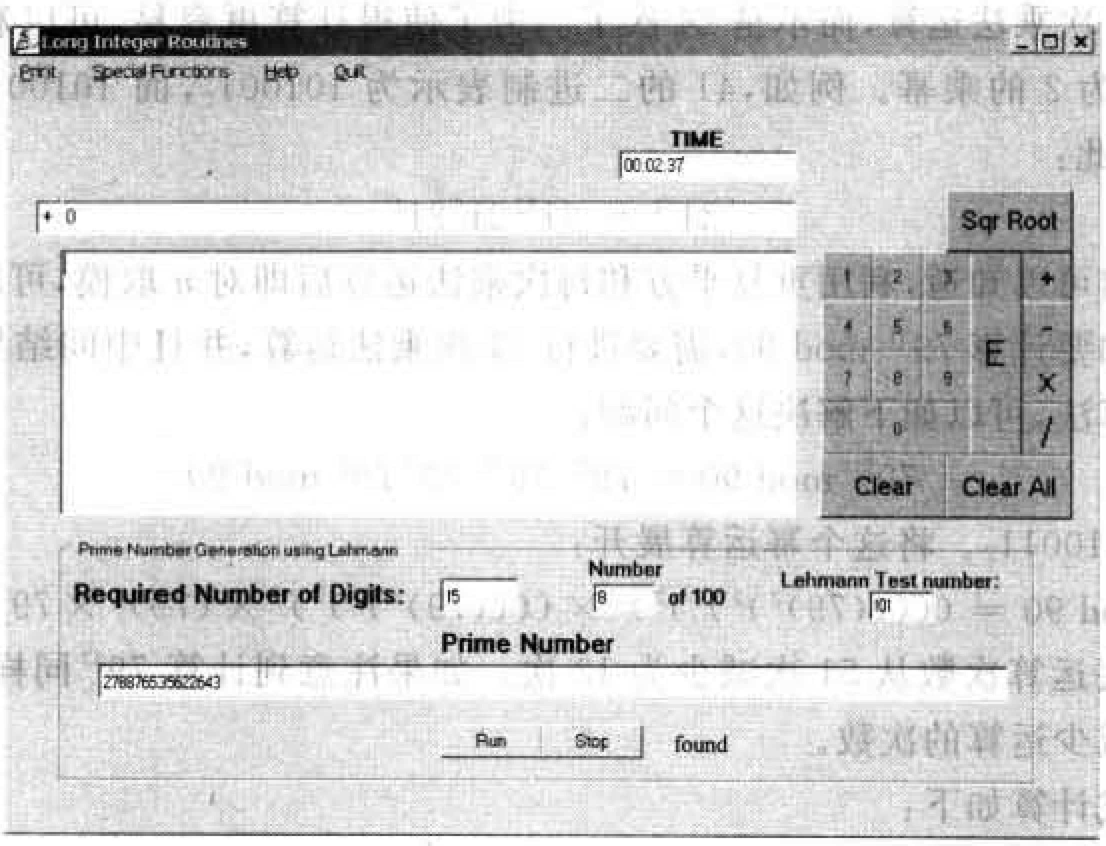


图 8.11 使用 CAP 软件查找素数

为 RSA 选取的素数不仅应足够大，以防止对 n 的因子分解攻击，而且两个素数之差也应足够大。如果 p 和 q 相差不大，那么就可以用 Fermat 算法对 n 进行因子分解。一个原则是较小的素数应为 n 的 40～45%。这就是说，如果 n 有 100 位，那么 p 应大约为 45 位，而 q 应大约为 55 位。

8.3.3 快速指数计算

除了找出大素数，进行模运算外，Bob 和 Alice 还面临使用 RSA 的另一个问题，他们还得计算大整数的幂。例如，如何快速且高效地计算：

$$33890123410097^{23015} \bmod 131$$

最简单的方法是将 33890123410097 自乘 23015 次，然后对 131 取模。记住，这可是将一个 14 位的数字进行 23000 多次乘法。用手工来完成是不可能的，甚至用计算机也是比较

慢的。幸运的是，Bob 和 Alice 有快速实现该计算的方法。

一个简单的方法是在每次乘法运算后就进行取模运算，而不是等到全部乘法运算完成后再进行。这样做是可以的，因为：

$$ab \bmod n = (a \bmod n)(b \bmod n) \bmod n$$

这样在每次取模运算后的数字就变小了，当乘法的次数仍是一样多。

事实证明，还可以用称为**重复平方**(repeated squaring)的方法来减少乘法的次数。例如，要计算 21^{16}，需要进行 16 次乘法，但由于 $16=2\times2\times2\times2$，因此可以改写为：

$$21^{16} = 21^{(2\times2\times2\times2)} = ((((21)^2)^2)^2)^2$$

从而可以减少为只有 4 次乘法的运算。这对所有指数为 2 的乘幂都是可以的。如果指数不是 2 的乘幂，那么就需要将它分解成 2 的乘幂之和。也就是说，如果指数为 24，那么就现将 24 分解为 2^4+2^3，因此：

$$21^{24} = 21^{(16+8)} = 21^{16}21^{8} = ((((21)^2)^2)^2)^2 \times (((21)^2)^2)^2$$

这就只需 8 次乘法运算，而不是 24 次了。为了使得计算更容易，可以利用指数的二进制表示，将它改为 2 的乘幂。例如，41 的二进制表示为 101001_2，而 101001 与十进制 $2^5+2^3+2^0$ 相等，因此：

$$21^{41} = 21^{32}21^{8}21^{1}$$

从上面介绍可以知道，利用重复平方和每次乘法运算后即对 n 取模，可以大大地减少计算工作量。例如要计算 $79^{51} \bmod 90$，需要进行 51 次乘法运算，并且中间结果非常大。但使用快速指数计算法，可以如下解决这个问题：

$$79^{51} \bmod 90 = 79^{32}\ 79^{16}\ 79^{2}\ 79^{1} \bmod 90$$

因为 $51=110011_2$。将这个幂运算展开：

$$79^{51} \bmod 90 = (((((79)^2)^2)^2)^2)^2 \times ((((79)^2)^2)^2)^2 \times (79)^2 \times 79 \bmod 90$$

这使得乘法运算次数从 51 次减少为 13 次。如果注意到计算 79^{32} 同样需要计算 79^{16}，还可以进一步减少运算的次数。

上面的等式计算如下：

$$79^2 \bmod 90 = 31$$

$$((79)^2)^2 \bmod 90 = (31)^2 \bmod 90 = 61$$

$$(((79)^2)^2)^2 \bmod 90 = (61)^2 \bmod 90 = 31$$

$$((((79)^2)^2)^2)^2 \bmod 90 = (31)^2 \bmod 90 = 61,\quad \text{这就是 } 79^{16} \bmod 90$$

$$(((((79)^2)^2)^2)^2)^2 \bmod 90 = (61)^2 \bmod 90 = 31,\quad \text{这就是 } 79^{32} \bmod 90$$

这样，问题就变成了 $31\times61\times31\times79 \bmod 90=19$ 了。

8.4 CAP 软件的 RSA 实现

利用 CAP 软件的奇数生成器，Bob 和 Alice 生成了两素数：

$$p = 232286714550223 \quad \text{和} \quad q = 816488284824217$$

利用 CAP 软件的长整数计算工具，可以得出 RSA 所需的其他数字：

$$N = 189659381150564068093773150391$$

且

$$(p-1)(q-1)=189659381150563019318773775952$$

他们选取一个素数作为 d 的值，该值比 p 大，且与 $(p-1)(q-1)$ 互质：

$$d=1175885676400129$$

利用 CAP 软件的反模运算，可以得出私钥：

$$e=53874618627671702415711710977$$

完成以上工作后，他们就可以开始使用 RSA 了。先输入明文，然后从 CAP 软件的 Cipher 菜单选取 RSA 菜单项。如图 8.12 所示，Bob 和 Alice 选择块的大小为 32 位，然后输入公钥。注意，加密一条含有 70 个字符的消息花费了差不多 2 分钟。性能是 RSA 的一个严重问题，下一节将详细介绍。

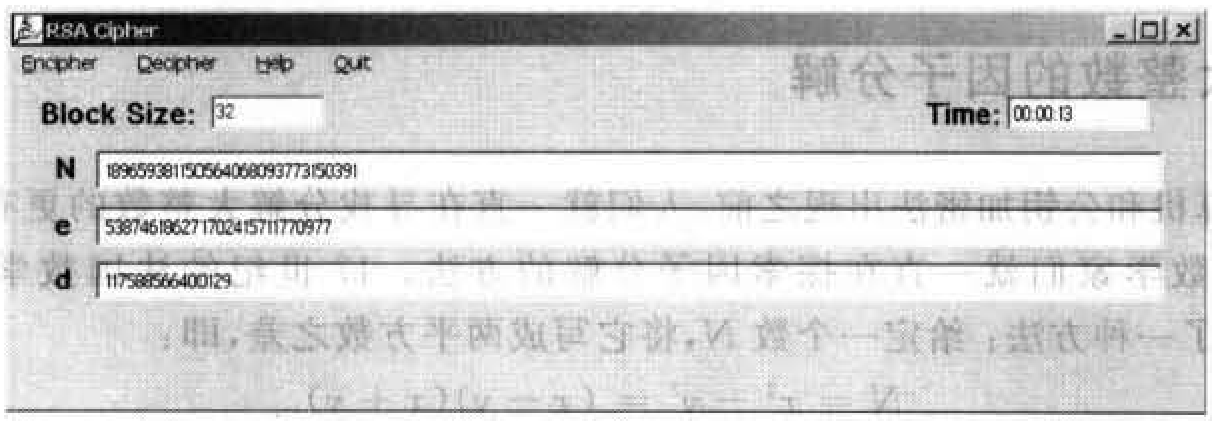


图 8.12　RSA Cipher 窗口

8.5　RSA 分析

在 Bob 和 Alice 首次尝试使用大密钥(其实并不够大)的 RSA 时，遇到了一些困难。首先，密钥本身很大，不可能记得住，要正确输入都困难。其次，该算法运行慢——大约是 DES 的千分之一。如果要使用一个理想的密钥(即密钥足够大)，这些问题就会变得更糟糕了。因此，RSA 很少用来加密明文。它的主要作用是为密钥交换和数字签名提供一种安全的加密方法(关于这些内容请见第 9 章的详细介绍)。

但是，作为在现实中曾经使用过的加密法，RSA 仍是重要的一种。Eve 怀疑 Bob 和 Alice 改用 RSA 了，于是开始花大力气去研究它，以期能发现它的一些弱点。

对 RSA 的首选和最明显的攻击方法是将公钥分解因子，以找出 p 和 q 这两个素数。分解因子最简单的方法是**尾部除法**(trial division)。Bob 和 Alice 就是用尾部除法来确定某个数是否为素数的。如果不为素数，那么该除法的结果是生成分解因子。

Eve 希望尾部除法能有所作为，她截获了 Bob 和 Alice 之间的一条消息：

872113	423723	1722277	382609	1166418
489217	1346472	478777	1262541	2436322
2523374	2700185	1767245	1602334	1722277
2867006	1275413	1760613	2682469	755208

Eve 查找 Bob 和 Alice 的公钥，发现了如下内容：

$$N=2949691\quad e=1801$$

Eve 所要做的是将 N 分解成两个素数 p 和 q。使用 CAP 软件，不到一秒钟的时间就发现其中一个因子为 1031。然后，利用乘法运算，就可以得出另一个因子为 2861。接下来，Eve 按照 Bob 和 Alice 计算私钥 d 的相同步骤，可以得出 d 为 201021。有了这个私钥，Eve 就可以将上面的密钥解密成如下明文："Alice I hope our key is large enough Bob"。当 Bob 和 Alice 发现他们的通信又一次被 Eve 阅读后，认为小密钥的 RSA 不再可靠了。当 Eve 检查 Bob 和 Alice 的新公钥时，发现该公钥大了许多：

$$N = 68969781959371 \quad e = 2623883$$

利用快速尾部除法，Eve 可以发现该新密钥的素数因子为 52013371 和 1326001，但是花费了近 20 分钟的时间。Eve 知道，Bob 和 Alice 最终会选用一个足够大的密钥，使尾部除法不再可用了，于是决定寻找另一种能将大整数分解因子的方法。

8.5.1 大整数的因子分解

早在计算机和公钥加密法出现之前，人们就一直在寻找分解大整数的更有效方法。几个世纪以来，数学家们就一直在探索因子分解的方法。17 世纪的法国数学家 Pierre de Fermat 提出了一种方法：给定一个数 N，将它写成两平方数之差，即：

$$N = x^2 - y^2 = (x - y)(x + y)$$

这就可以立即得出 N 的两个因子。分解 N 的问题也就变成了查找 x 和 y 值了。该算法是先猜测一个 x 值，然后用 $y^2 = x^2 - N$ 计算出 y 的值。由于 x 必须大于 N 的平方根，因此首先猜测必须是 $x = \text{sqrt}(N) + 1$，然后再用它来计算 $z = x^2 - N$。如果 z 是某个数的平方，那么查找结束；否则，尝试下一个值 $x = \text{sqrt}(N) + 2$。继续该过程，直到找到分解因子或时间用完为止。CAP 软件实现了这种方法，从 Analysis 菜单的 Public Key 中单击 Factor 按钮即可。这种方法不仅比尾部除法更快，在分解因子大小相同的情况下，这种方法运行最快。(还记得在 8.3 节末尾介绍的选择素数的指导原则吗?)在这种情况下，分解因子接近于 N 的平方根，因此查找的范围最小。图 8.13 是 CAP 软件的一个运行示例，它表明，用 Fermat 将 6664407857 因子分解花费了近 7 分钟，而用同样的程序分解 268380942439(光分解出的两个数字都比前面示例的要大)，CAP 只花了 2 秒钟。

$$268380942439 = 511897 \times 524287$$

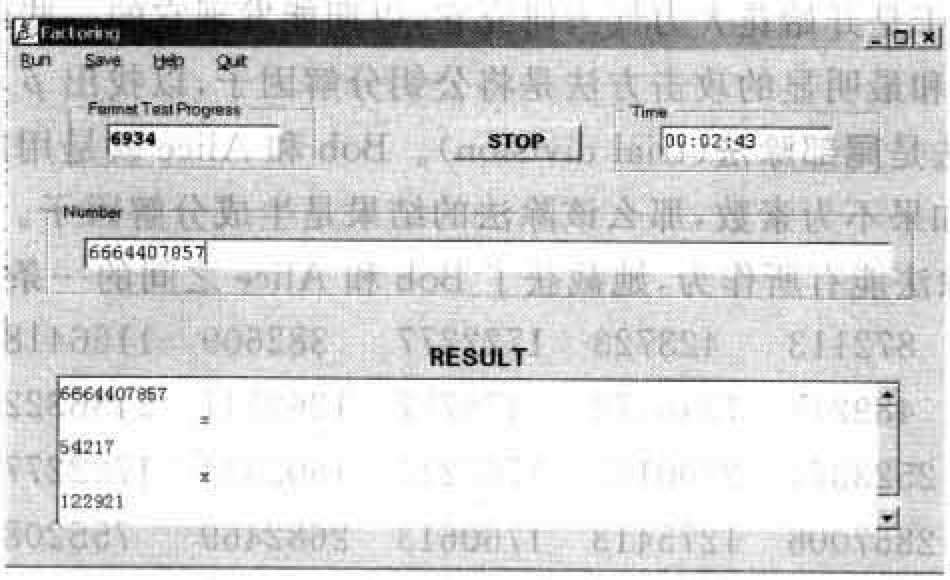


图 8.13 CAP 软件的 Fermat 方法运行示例

注意，这两个素数因子相当接近。事实上，RSA 建议两个素数 p 和 q 应相差大一些，以避免因子分解攻击法。

J. W. Pollard 开发了多种因子分解算法，这些算法并不要求有接近的因子。其中有一种称为 Pollard Rho 因子分解算法。该算法是查找两个这样的数：对 n 的一个素数因子取模后的结果相等。这听起来很复杂，其实相当简单。设要分解因子的数为 n，p 为 n 的一个未知因子，以及 $x_0, x_1, \cdots, x_m$，其中 $x_{j+1} = f(x_j) \bmod n$，数列 $z_k = x_k \pmod p$。由于数列 z 的值只限于 $0,1,\cdots,(p-1)$，因此它比 x 数列(即 $0,1,\cdots,(n-1)$)重复得更快，如果 z 数列出现重复(即 $z_i = z_j$)，这就意味着 $x_i = x_j \pmod p$。这又反过来说明 p 可以除尽 $x_i - x_j$，从而表明 p 可除尽 n，因此 p 也必定能除尽 $\mathrm{GCD}(x_i - x_j, n)$。如果 $\mathrm{GCD}(x_i - x_j, n)$ 不为 1 或 n，那么它就是 n 的一个因子。具体实现算法如下：

> Pollard Rho 因子分解算法
> 输入一个合数 n
> 输出 n 的一个因子 d
> 1. 设 $a=2, b=2$；
> 2. For $i=1,2,\cdots$ do
>   2(a) $a=a^2+1 (\bmod n)$，　$b=b^2+1 (\bmod n)$；
>   2(b) $d=\mathrm{GCD}(a-b, n)$；
>   2(c) if $1<d<n$，那么就停止并返回 d；
>   2(d) if $d=n$，那么也将停止，该算法失败。

1980 年用该算法将第 8 个 Fermat 数进行了因子分解：

$$2^{2^8}+1 = 1238926361552897 \times p62$$

其中，$p62$ 是一个 62 位的整数。

CAP 软件实现了该算法。在 Public Key 选项下可运行 Pollard Rho 算法。如图 8.14 所示，用 Pollard Rho 算法分解 6664407857 仅用了 2 秒钟。(用 Fermat 算法花费了近 7 分钟。)

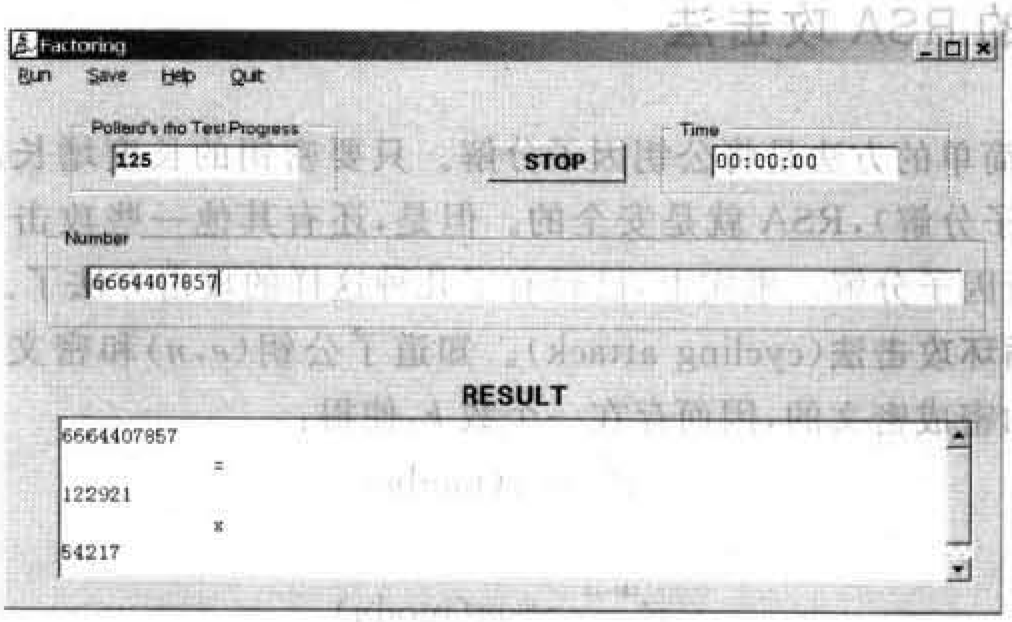


图 8.14　CAP 软件的 Pollard Rho 方法运行示例

CAP 软件还实现了另外两种常用的因子分解算法：Pollard 的 $p-1$ 和连续因子分析法(有关这些算法的详细内容，请参加数论方面的教材)。除了 CAP 软件提供的这 4 种方法之

外，现今使用的其他因子分解算法还有**二次方筛选法**(quadratic sieve)和**数字域筛选法**(number field sieve)。

为了应对更快的计算机和新的因子分解算法，RSA不断要求素数生成器生成更大的素数。在1977年的《科学美国人》专栏中，Martin Gardner为第一个分解RSA-129的人设立一个100美元的奖。这个129位的数字是：

$$N = 114381625757888867669235779976146612010218\\ 2967212423625625618429357069352457338978305\\ 97123563958705058989075147599290026879543541$$

1994年，Derek Atkins宣布成功地将它因子分解了：

$$\text{RSA-129} = 3490529510847650949147849619903834177646384\\ 93387843990820577 \times 32769132993266709549961\\ 988190834461413177642967992942539798288533$$

该工作是用二次方筛选算法来完成的。该算法是运行在Internet中多个志愿者的计算机上的，利用他们的空闲时间。这大约有1600台计算机参与，花费了8个月才完成该因子分解工作。

RSA公司为第一个成功分解一系列数字的人提供了现金奖励。1996年4月，有人用数字域筛选算法将RSA-130成功分解了。完成该工作使用了300台工作站和PC机，还有一台Cray超级计算机，花费了7个月的时间。2003年2月，还未被分解的最小的RSA素数为174位：

$$18819881292060796383869723946165043980716356337941738270076\\ 33564229888597152346654853190606065047430453173880113033967\\ 161996923212057340318795506569962213051687593076502570 59$$

第一个成功分解它的人可获10 000美元。这种挑战性的素数已达617位了(奖金为200 000美元)。

8.5.2 其他的RSA攻击法

攻击RSA最简单的方法是将公钥因子分解。只要密钥的长度增长超过因子分解技术(即无法将密钥因子分解)，RSA就是安全的。但是，还有其他一些攻击RSA的方法，这些方法并不需要进行因子分解。事实上，已经有了几种这样的攻击方法了。

一种方法是**循环攻击法**(cycling attack)。知道了公钥(e,n)和密文c，且知道是用$c=m^e \bmod n$将明文加密成密文的，因而存在一个数k，使得：

$$c^{e^k} = c(\bmod n)$$

这意味着：

$$c^{e^{(k-1)}} = m(\bmod n)$$

该攻击法需要计算：

$$c^e,\quad c^{e^2},\quad c^{e^3},\cdots$$

并将以上数字对n取模，直到某个结果为c为止，此前一个就是明文。对于一个大数n，这种攻击法需要的工作量与因子分解的差不多，因此它并不是一种能对RSA构成很大威

胁的攻击法。这就是说，除法有人能提出一种不需要穷尽查找就能找到 k 的方法，否则 RSA 就是安全的。

RSA 并不具备**语义安全性**(semantic security)。也就是说，给定一个可能的已知消息集 $m_0,m_1,\cdots,m_n$，以及相应的一个密文 c_i，有可能找出该 i 的值，即能知道是这几个消息中的哪一个加密成密文 c_i 的。这很容易，只要用公钥将所有 n 个消息加密，然后将结果密文与密文 c_i 进行即可。但是，就目前来说，循环攻击法和 RSA 缺乏语义安全性都不是很严重的问题，还有其他一些攻击法更值得我们关注。

8.5.2.1　旁道攻击法

正如 DES 一样，RSA 也能用**旁道**(side-channel)攻击法来攻击。这种攻击法充分利用了 RSA 加密法实现时可能产生的信息泄漏。据关于这种攻击法的最早报道说，该攻击法是利用了 Hagelin 加密机所发出的声音。Hagelin 加密机实现的是一种类似于 Engima 的旋转轮加密法。当旋转轮转动时，会发出滴答声。通过监视其声音，有可能推断出旋转轮的设置。

一种更为现代的旁道攻击法是**时耗密码分析法**(timing cryptanalysis)，由 Paul Kocher 于 1995 年 11 月提出的。时耗密码分析法要求攻击者能计算出加密-解密多个消息所耗费的时间。有了这个时耗结果，攻击者就可以推断出某个指数位是 1 还是 0，因为处理指数位为 1 的消息耗时比为 0 的更长。通过这种方法猜测出的每个位，并用对多个消息所猜测的平均值与完成加密-解密的时间进行比较，攻击者就可以找出保密密钥。

有多种防止或使时耗攻击法失效的对策。一种比较明显的方法是在 RSA 的实现中添加随机延时。这可能有损于性能，因为 RSA 已经相当慢了，因而可能不是一种能接受的解决方法。另一种方法是使用"隔障法"，它引入了两个额外的除法操作，从而隐藏了实际的时耗。用于解密算法的隔障处理过程包括：

(1) 选取一个随机数 r；

(2) 计算 $m=r^{-1}(cr^e)^d \bmod n$(而不是 $m=c^d \bmod n$)。

这样做是可行的，因为：

$$m = r^{-1}(cr^e)^d \bmod n = (c^d \bmod n)(r^{-1})(r^{ed} \bmod n) = (cd \bmod n)(r^{-1}r^1 \bmod n) = c^d \bmod n$$

RSA 数据安全公司(该公司拥有 RSA 的专利权)在其实现中使用了隔障法，并发现给整个算法增加了 5%的延时。

另一种类似于时耗攻击法的旁道攻击法是，测量实现 RSA 加密法的设备的能量消耗。如图 8.15 所示，每个操作所耗费的能量不同。通过监视能量的消耗情况，就有可能确定保密密钥 d 的每个位，因为如果某位为 1，就多一个乘法运算。图 8.16 为消耗能量分析的一个示例。

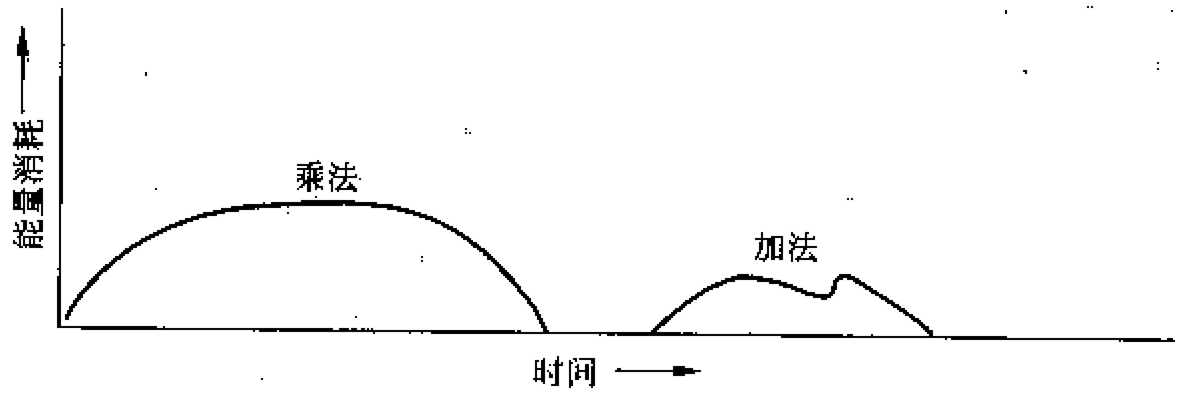


图 8.15　RSA 运算的能量消耗

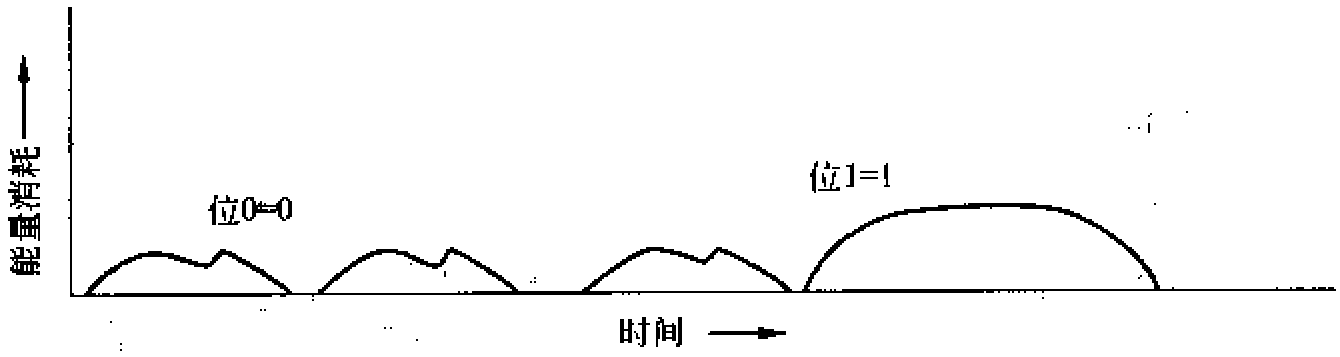


图 8.16 还原 d 的各个位

防止这种攻击法的一种对策是，应对实现该加密算法的软件指令进行仔细分析，尽量避免使用条件分支。

8.5.2.2 RSA 使用不恰当攻击法

有一些攻击法利用 RSA 系统的不恰当使用进行攻击。这些不恰当使用包括公钥和私钥的选用不当，用类似的公钥加密同一消息等。

如果私钥 d 太小，那么从公钥(e,n)中就可以推断出它。由于 RSA 需要很多计算量，因此有时也会使用小的私钥 d。较小的私钥 d 可以减少计算量，从而就减少了功耗，使得 RSA 对低功耗设备更为实用。RSA 不恰当使用攻击法最初是由 M. Wiener 提出的，Boneh 和 Durfee 对它进行了进一步发展，他们发现，如果 $d<n^{0.295}$，该攻击法就可能发现 d。因此，根据推荐使用的要求，如果 n 为 1024 位长，那么 d 至少应为 300 位长。

部分密钥暴露攻击法(partial-key-exposure attack)是基于小公钥值 e 的。由于选用的私钥用 d 较小，因而也可能选用较小的公钥 e。e 的值越小，加密时所需的乘法运算就越少，因此加密过程也就越快。e 的最小可能值为 3，但出于安全因素，通常选用 65337。e 的这种值(不是大随机数 e)将可使加密处理过程中的乘法运算次数从 1000 次减少至大约 20 次。另一方面，根据 Boneh、Durfee 和 Frankel 的研究，如果 n 为 k 位，且如果 d 的最少有效位$(k/4)$已知，那么找出所有的 d 所花时间与$[e \log (e)]$成线性关系。因此，如果 e 较小，那么就可以较快地找出 d 的 1/4 位，从而能得出整个私钥。

公共模数攻击法(common modulus attack)也是不恰当使用 RSA 算法的结果。由于要寻找能保证 n 安全的大素数很耗时间，某些组织可能会先生成一个值 n，然后给每个用户不同的(d,e)对。这是基于这样一个假设的：Bob 没有 Alice 的私钥，就不能阅读发送给 Alice 的消息。这是错误的，因为 Bob 利用它的公钥和私钥可以将 n 分解因子。一旦分解后，从 Alice 的公钥中，Bob 就可以发现 Alice 的私钥。

外部攻击者(如 Eve)并不像 Bob 那样也具有分配的密钥对(d,e)，但仍可破解加密消息，因此这个问题就更糟糕了。如果 Alice 给 Bob 和 Paul 发送一个消息 M(记住，这是在一个组织中，所有的公钥有一个公共模数 n)，她就需要找出 Bob 的公钥 e_1 和 Paul 的公钥 e_2。利用这些密钥，Alice 就可以生成两个不同的密文(一个给 Bob，另一个给 Paul)：

$$C_b = M^{e_1} \bmod n \quad (\text{给 Bob 的})$$

$$C_p = M^{e_2} \bmod n \quad (\text{给 Paul 的})$$

如果 Eve 能得到这两个密文，再加上知道这两个公钥有最大公约数 1(即 $GCD(e_1,e_2)=1$)，那么 Eve 所要做的就是找出两个数 x 和 y，使得：

$$xe_1 + ye_2 = 1$$

有了 x 和 y，Eve 就可以还原消息 M：

$$C_b^x C_p^y = (M^{e_1})^x (M^{e_2})^y = M^{xe_1+ye_2} = M^1 = M$$

因此，永远不要使用有不同指数 e 的相同模数 n。

8.6 ElGamal 公钥系统

RSA 是靠因子分解大数的困难来保证其安全性的。还有其他的一些公钥系统，它们是靠其他类型的运算操作困难度来保证其安全性的。其中，ElGamal 加密法是 1985 年由 T. ElGamal 开发的。这样加密法是基于求解离散对数问题的困难度的。

离散对数问题首先给定一个素数 p，两个整数 a 和 b，且对于某些整数 x 有 $b=a^x \bmod p$ 成立。该问题是要找出一个 x，也就是说，求解方程 $x=\log_a b$。如果 p 较小，该问题就可以用穷尽查找法来解决。例如，假设 $p=17, a=3, b=5$，求解满足 $5=3^x \bmod p$ 的 x 值。从 1 开始，尝试所有可能的值，直到找到一个满足该等式的结果：

x	1	2	3	4	5	
3^x	3	9	10	13	5	mod 17

因此，$x=5$。只要 p 比较小，用穷尽查找法就可以很容易求解该问题。如果 p 比较大，假设是一个 100 位的素数；那么就像整数的因子分解问题那样很难求解了。

与 RSA 一样，ElGamal 系统也是一个公钥算法，因此也是要公布一些密钥数字，其他的保密数字则用于加密。密钥是通过选择一个大素数 p 来生成的。通常建议 $p-1$ 能被另一个大素数整除。计算出一个生成器数字 g（稍后将详细介绍），并选择一个比 $p-1$ 小的随机数 a。有了这些数字后，计算 $b=g^a \pmod p$。公钥是由 (p,g,b) 这三个数字组成的。要找出给定公钥的私钥 a，攻击者必须要能求解这个离散对数问题。

如果 Bob 要发送一条消息给 Alice，可先找到 Alice 的公钥 (p,g,b)，并用整数 $m(0\leqslant m\leqslant p-1)$ 表示该消息，然后选用一个随机密钥 k（小于 $p-1$）。有了这些数字后，Bob 可计算以下两个数：

$$C_1 = g^k$$

和

$$C_2 = mb^k$$

Bob 就可以将密文 $C=(C_1, C_2)$ 发送给 Alice。

当 Alice 接收到该密文后，利用私钥 a，通过以下计算就可以还原明文：

$$m = C_2 C_1^{-a} \bmod p$$

上式是成立的，因为：

$$C_2 C_1^{-a} = mb^k (g^k)^{-a} = m(g^a)^k (g^k)^{-a} = mg^{ak} g^{-ak} = m \bmod p$$

例如，如果 Alice 选择的初始素数 $p=11$（对真正的安全性来说，这个数太小了），找一个素数 $g=7$，选用的随机私钥 $a=2$。于是可计算公钥 $b=7^2 \bmod 11=5$。Alice 公布她的公钥 (p,g,b) 为 $(11,7,5)$。但将 $a=2$ 保密。如果 Bob 要给 Alice 发送字母“a”，首先将该字母分

解成一系列的数字，其中每个数字都应小于 11。由于字母“a”的 ASCII 表示为 0110 0001，因而将它分解成 4 个消息(01 10 00 01)或十进制的(1,2,0,1)。接下来，选一个随机数 $k=3$，如下计算并将结果发送给 Alice：

m	c_1	c_2
1	$7^3 \bmod 11=2$	$1\times5^3 \bmod 11=4$
2	$7^3 \bmod 11=2$	$2\times5^3 \bmod 11=8$
0	$7^3 \bmod 11=2$	$0\times5^3 \bmod 11=0$
1	$7^3 \bmod 11=2$	$1\times5^3 \bmod 11=4$

即所生成的密文为[(2,4) (2,8) (2,0) (2,4)]。

当 Alice 从 Bob 那里接收到该消息后，利用私钥 $a=2$ 进行如下计算：

$$(2,4)\ m = 4(2)^{-2} = 4(4)^{-1} = 12 \bmod 11 = 1$$

$$(2,8)\ m = 8(2)^{-2} = 8(4)^{-1} = 24 \bmod 11 = 2$$

$$(2,0)\ m = 0(2)^{-2} = 0(4)^{-1} = 0 \bmod 11 = 0$$

$$(2,4)\ m = 1$$

Alice 就可以将该消息重新组装成字母“a”了。

这的确行的通，但 Bob 犯了一个致命的错误。他对每个消息 m 都使用同一个随机值 k。在这种情况下，如果 Eve 猜测出了第一个消息数字，那么她就可以找出 b^k(因为 $c_2=m_1b^k$)，因此 $b^k=c_2/m_1$。利用 b^k，她就可以得出所有其他的消息数字了。在上面的示例中，如果 Eve 猜测出了第一个数字为 1，那么她就可以知道 $b^k=4$。由于她还知道消息 2 的密文为(2,8)以及关系式 $c_2/m_1=b^k$，于是：

$$m_2 = 8(4)^{-1} \bmod 11 = 2$$

因此，使用 ElGamal 的一条原则是，对每个消息块，应使用不同的随机数字 k。

8.6.1 生成器数字

Bob 和 Alice 在使用 ElGamal 系统中面临的惟一问题是，找一个生成器数字，该数字与他们选用的素数相关联。事实上，没有必要真正地使用生成器数字，很多数字都是可以的。但是，如果选择不当，将会使系统更容易攻破。

不幸的是，并没有现成的方法能为给定的素数 p 计算出一个生成器数字，除非已经知道了$(p-1)$的因子分解情况。如果知道了$(p-1)$的分解因子为 $q_1,q_2,\cdots,q_r$，那么就可以用如下等式来计算 p 的生成器数字 g：

$$g^{(p-1)/q_i} \bmod p \neq 1$$

当然，这种方法的问题在于，要将$(p-1)$这样的大素数因子分解是很困难的。因此，要寻找素数 p 及其生成器数字 g 的最好方法是，首先构建已知其因子分解形式的$(p-1)$。

实现这种方法的步骤如下：

1. 选择一个随机数 n，它将确定$(p-1)$的素数因子的个数；
2. 选择 n 个随机数 $q_1,q_2,\cdots,q_n$(应验证以确保它们都是素数)；
3. 选取 $n+1$ 个随机数 $e_1,e_2,\cdots,e_n$ 作为指数；

4. 如下定义素数 p：

$$p = (2^{e_0} \times q_1^{e_1} \times q_2^{e_2} \times \cdots \times q_n^{e_n}) + 1$$

5. 验证 p 的确是素数。

如果 p 为素数，那么我们就有了$(p-1)$的分解因子，从而可用它们来得出生成器数字。

8.6.2　用 CAP 软件来实现 ElGamal

CAP 软件提供了一个 ElGamal 版本，包括一个选取密钥的例程。要访问该例程，选择 Cipher 菜单下的 ElGamal 项，弹出 ElGamal 窗口。在 ElGamal 窗口中，选取 Key 菜单下的 Create 项，将弹出 Create Key 窗口。在该窗口中输入$(p-1)$的素数因子个数和大小，选取 Create 菜单下的 Find p and q 项，就可以生成你所需的 ElGamal 密钥。CAP 软件可以找出指定大小的随机素数、每个素数的随机指数，并将它们相乘以得到$(p-1)$。接着，使用 Miller-Rabin 测试来确定 p 是否为素数。还可以利用$(p-1)$的分解因子来得出生成器数字 g。整个过程可能需要花费一定的时间。一旦完成，就可以输入一个保密数字，选取 Create 菜单下的 Find b 项，即可完成整个处理过程。注意记得用 Create 菜单下的 Save 项来保持这些密钥数字。图 8.17 为一个示例，它花费了 8 分多钟才得到如下公钥：

$$p = 39823489096253$$
$$g = 874184$$
$$b = 10648334133598\text{(保密密钥为 14679)}$$



图 8.17　Create ElGamal Key 窗口

CAP 软件可以使用这些密钥数字来加密-解密 32 位长的消息块(实际的块大小取决于 p 的大小)。例如，使用前面介绍的公钥，CAP 软件可以将“This is a test of elgamal”消息加密成如下的数字集：

4134442763715

18825904687947

27237957908646

19645981685387

16917762883314

3209378757176
25691232474630
1213006145262
7993144971816
3601640567706
2902376418316
17812051359532
2275290146625
38479927281019

8.6.3 小结

ElGamal 加密法引入了一些新的特征，这些特征是到目前为止所介绍过的其他加密法未曾出现过的。最显著的一个特征是密文生成中随机数的引入。这意味着相同的明文可能生成不同的密文，这从安全的角度来说是一个真正的提高。但是，由于密文是明文的两倍大，从而增加了对存储空间的需求，降低了数据传输的速度。

8.7 背包加密法

1976 年，Ralph Merkle 和 Martin Hellman 首先提出了公钥加密的可能性（20 世纪 60 年代，这一概念就已经为英国的情报机构所熟悉了，但在 90 年代以前一直出于保密状态。）。1978 年，他们基于背包问题实现了一个公钥加密系统。最初的背包加密法已被破解了，因此今天已不再使用了（尽管后来进行了修改并去除了这个问题）。但是，该加密法还是很有趣的，因为它在加密技术中采用了 NP 完全问题。

背包问题可以很容易形象地描述。设一个体积为 t 的背包，以及一个元素集 $1,\cdots,n$，其体积分别为 $t_1,\cdots,t_n$，是否存在这样一个子集：其体积之和等于背包的体积？例如，假设一个背包的体积为 20，元素集有 7 个元素，每个元素的体积分别为 5、3、7、1、12、15 和 19。哪些元素可以填满该背包。在这个示例中，有多个正确答案。第 4 和 7 个元素之和等于 40，第 1 和 6 个元素之和也等于 20。

背包问题更正式的数学定义是，给定一个数集 $m_1,\cdots,m_n$，一个目标总和 S，是否能找出 $b_1,\cdots,b_n$（其中 b_i 为 0 或 1），使得：

$$S = b_1m_1 + b_2m_2 + \cdots + b_n m_n$$

对背包问题而言，S 即为总体积，如果 b_i 为 1，那么第 i 个元素就在背包中。

Merkle-Hellman 背包加密法就是以背包问题的形式来加密消息的。其背包以一系列的数字 $m_1,\cdots,m_n$ 给定，消息块是 n 个二进制位 $b_1,\cdots,b_n$ 的集合，密文就是前面给出的总和 S。例如，给定一个背包列表为(5,14,9,23,16,7,31 27)，字符"a"用 ASCII 码表示为 0110 0001，那么其密文就为：

$$S = (0\times5)+(1\times14)+(1\times9)+(0\times23)+(0\times16)+(0\times7)+(0\times31)+(1\times27)=50$$

反过来，知道了密文(在上面的示例中为 50)，通过求解背包问题就可以还原明文。在上面的示例中，用穷尽查找法还原明文也不太困难。由于其背包只有 8 个数字，对 b 而言，只有 $2^8=256$ 种 0 和 1 的组合。但是，如果背包含有 100 个元素，那么情况又会如何呢？这将有 2^{100} 种可能的组合。假设每秒钟检测一百万种组合，那么要找出一个正确的组合，平均得花费 5×10^{25} 秒的时间，大约等于 100 年。

背包问题所蕴含的思想是，利用寻找总和困难这一事实作为包含信息加密的一种方法。当然，合法用户必须能在正常的时间内解密信息，因此还必须具有解密算法，且只有合法用户才能使用。这种解密算法是基于一个简单的背包问题的。简单的背包具有一系列的数字，其中的每个数字都比其前面所有数字之和要大。(3,5,9,18,38,75,155,312)就是这样的一个数字系列。注意，9 比 3+5 大，18 比 3+5+9 大，等等。要加密字符“R”，首先将它用 ASCII 码表示，即为 0101 0010，因此目标总和为 5+18+155(即 178)。对于这种形式的背包，从总和中可以很容易还原数列中的各个元素，从而很容易还原字符“R”。在上面的示例中，312 不可能参与总和的计算，因为目标总和比 312 要小。而 155 必须参与总和的计算，因为 155 之前所有数字之和只有 148，比 178 小。知道了 155 是目标总和 178 的一部分，就可以将它从总和中减去，即 178－155＝23。现在，很显然，75 和 38 都不可能参与总和的计算，因为它们都比 23 大。因此，18 肯定是目标总和的组成部分。同样，23－18＝5，这必定就是参与总和 178 的最后一个数了。从而可以知道，被加密字符的 ASCII 码为 0101 0010，即为字符“R”。

求解上面这种背包(即找出 $b_1,\cdots,b_n$ 的值)的算法如下，其中数字($m_1,\cdots,m_n$)和目标总和 S 为已知的：

```
for i=n downto 1 do
    if S>=m_i then
        b_i=1 and S=S-m_i
    else
        b_i=0
```

这种背包加密法很容易破解，这也就意味着它并不能保证数据安全。如果知道了背包的各个元素，那么从背包的目标总和就可以还原密钥的位。Merkle 和 Hellman 建议将这种简单的背包转换成 trapdoor 背包，它不是递增的，因而更难以破解。设有一个简单的背包数列 $A'=(a'_1,\cdots,a'_n)$，要转换成 trapdoor 背包数列的步骤如下：

(1) 选取一个整数 $u>2a'_n$；

(2) 选取另一个整数 w，且 $\mathrm{GCD}(u,w)=1$；

(3) 计算 $w \bmod u=1$ 的逆；

(4) 构建 trapdoor 背包数列 $A=wA' \bmod u$。

例如，在前面的示例中，

(1) 选取 $u>624$，假设为 672；

(2) 选取 $w=13$；

(3) $w \bmod u=1$ 的逆为 517(因为 $517\times13=1 \bmod 672$)；

(4) 构建 trapdoor 数列(39 65 117 234 494 303 671 24)。

利用 trapdoor 数列数量，字符"R"就成了 65＋234＋671＝970。因而光知道 trapdoor 数列很难解密。但是，如果知道了 w^{-1} 的值和原始递增数列，同样很容易解码。通过计算 trapdoor 数列的总和与 w^{-1} 的乘积，然后对 n 取模(即 517×970 mod 672＝178)，就可以计算出递增数列的目标总和。从递增数列中查找组成总和为 178 的各个元素，就可以还原字符"R"。这种加密法的保密密钥包括递增数列、u、w 和 w^{-1} 的值。trapdoor 数列作为公钥发布。

8.7.1 破解背包加密法

背包加密法不仅是首先使用公钥的加密法，而且也是第一个被严重泄密的加密法。有多种攻击法可以破解背包加密法的最初版本，从而促使其进行了修订，但背包加密法从没有被广泛使用。1983 年，Shamir 发布了他的背包加密攻击法。该方法无须寻找背包问题的求解，而是寻找重构递增背包的方法即可。我们应从中吸取的教训是，尽管某个加密法是受某些难题的保护，但这并不意味着，不能求解该难题就无法破解该加密法。

破解背包加密法的一种攻击方法是直接求解背包问题，包括使用遗传算法来进行可能求解的查找。

遗传算法是作为对进化式程序设计的修订而由 John Holland 提出的。其思想是基于生物学的自然选择模型来构建的一种查找算法。该结果是一个有向的随机查找过程。该过程首先构建所有可能求解的随机分布。这种分布可用来生成新的可能求解，而新的可能求解又可生成下一个可能求解，以此类推。当前可能求解中的最佳元素用来生成新的可能求解，从而使新的可能求解比前面的包含范围更小。事实证明，这在许多应用中是很有效的。

在人类的基因中，有三种过程并行地用于实现从一代到下一代的转变中。它们是选择、匹配和变异。基本的 GA 循环就是基于这三个过程的，如图 8.18 所示。

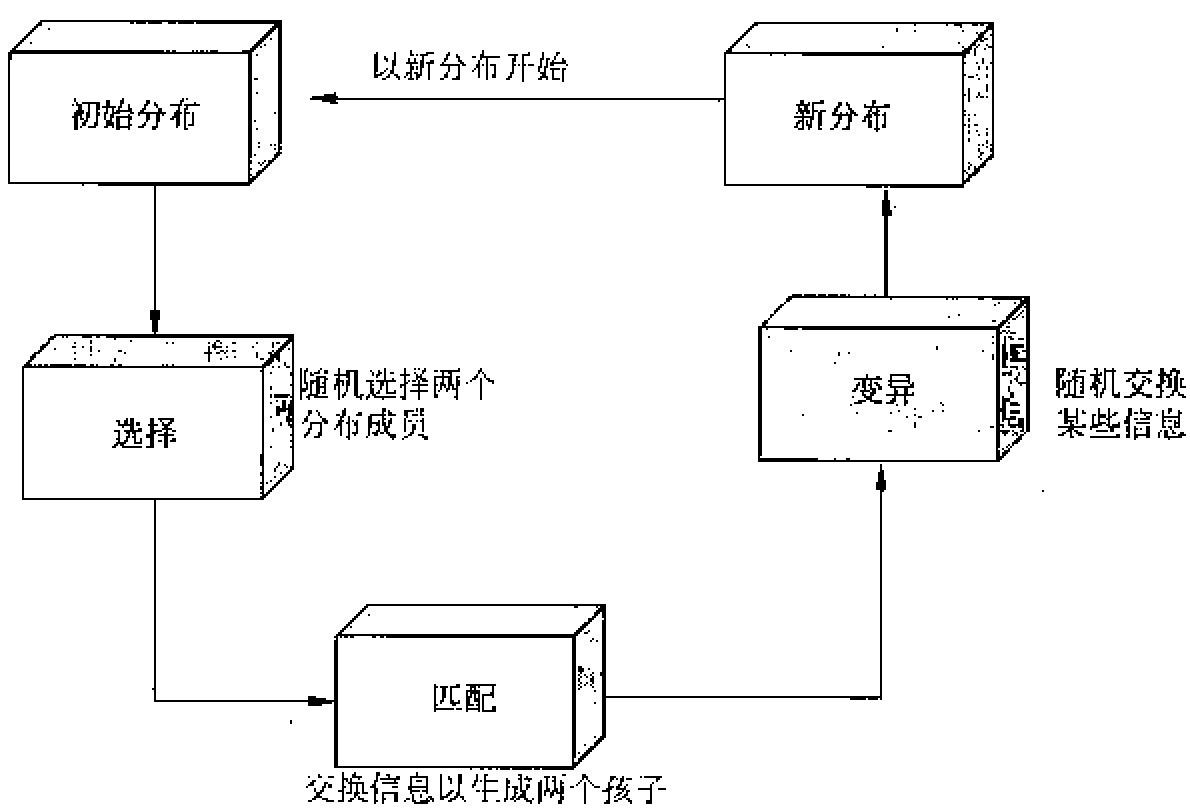


图 8.18 一个 GA 循环

个体是由二进制字符串集组成的。每个二进制字符串称为一个染色体，它是当前问题的一个求解。第一个基因算法步骤是将选择过程应用于该个体。选择过程决定这一代中的哪个字符串用于生成下一代。这可用偏差随机选择算法来完成。也就是说，从当前一代中随机选择双亲，使最好的字符串具有最大的被选概率。

第二步是匹配过程。它将确定下一代中二进制字符串的实际内容。在 GA 中最常用的方法称为**交叉**(crossover)过程。如果双亲字符串的长度为 r，那么所选择的随机数为 $1\sim r-1$，假设为 s，交叉过程就是将第一个双亲的 $s+1\sim r$ 位与第二个双亲的 $s+1\sim r$ 位进行交换，从而生成两个新的孩子，如图 8.19 所示。

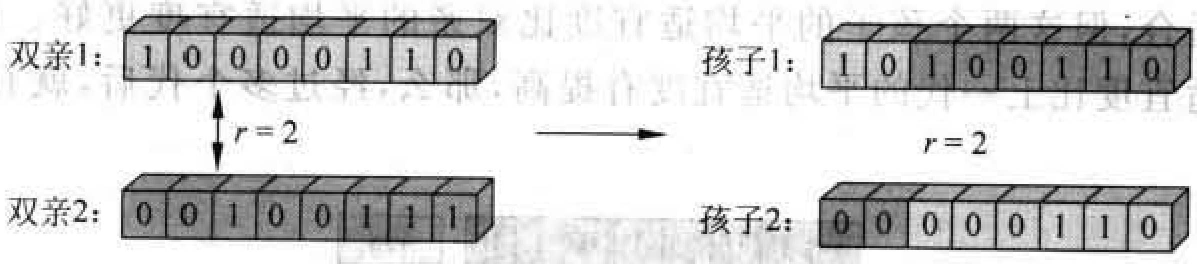


图 8.19　一个交叉操作

最后一步是变异。在算法的开始处，先设置一个固定的小变异概率。所有新字符串中的位都将基于该变异概率发生改变。如果变异概率为 0.01，那么每个位发生变异的概率为 1/100。重复以上三个步骤以生成每个新一代。不断以这种方式进行直到到达某个停止条件(例如，代的最大数)。

背包问题的表示结构形式很容易生成，因为该问题本身就有所暗示。表示加法的二进制位模式大概就是最好的表示结构形式了。例如，假设一个有 8 个元素的背包为(39 65 117 234 494 303 671 24)，目标总和为 970。GA 的一个常见总体元素为：

10101010 → 加法项为 1,3,5,7

如果要为遗传算法选择一个表示方案，那还有必要给定一个评估函数。该函数用来确定"最好"的表示形式。对于背包加密法，评估函数的基本结构很容易确定。它应计算某些给定项之和与该背包的目标总和相差多少。在这条通用的指导原则下，可以有很多可能的变量。背包问题一个很明显的函数是计算目标总和与当前总体元素项之和的差。在上面有 8 个元素的背包示例中，总体元素(1010101)之和与目标总和差为 $|(39+117+494+671)-970|=351$。

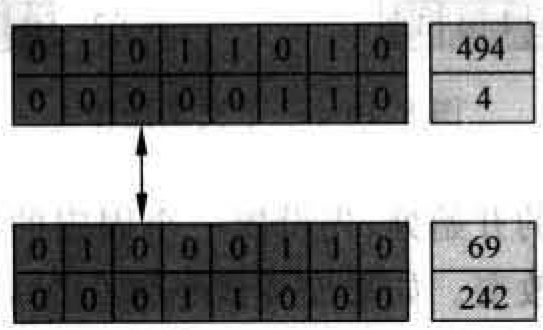
GA 最初的总体元素是随机生成的，并确定每个元素的值。假设随机生成了含有 6 个元素的总体元素，并计算出总体元素之和，如图 8.20 所示。给定一个染色体的总体元素，每个都有一个相应的值，遗传算法先随机选择两个元素进行匹配。具有更高适宜值

	分布								选中
1	1	0	1	1	0	0	1	1	115
2	1	1	0	0	0	0	0	1	842
3	0	1	0	1	1	0	1	0	494
4	0	0	0	0	0	1	1	0	4
5	1	1	0	0	0	1	0	1	539
6	1	0	0	0	0	1	1	0	43

图 8.20　总体样本

的染色体,被选择用来为下一代生成孩子的机会就更高。例如,对于上面的示例,适宜值越小,该元素被选取的概率越高。在初始的随机总体元素中,元素 4 和 6 是最好的,因此它们应该最有可能作为下一代的双亲。元素 2 就几乎没有机会将它的"基因结构"传递给下一代。

使用偏差随机选择方法(这样最好的元素最有机会被选取),假设元素 4 和 3 被选取。在 1~7 之间随机产生一个数——假设为 3,将这两个元素的 4~8 位相互交换,即可生成其孩子。匹配过程是作用于双亲的,因此,假设随机选择第二个孩子的位 7 进行补位(在本示例中将 1 变为 0)。这两步的操作结构如图 8.21 所示。尽管每个孩子都不如双亲的第二个元素适合,但这两个孩子的平均适宜度比双亲的平均适宜度更好。这样,只要下一代的平均适宜度比上一代的平均适宜度有提高,那么,经过多个代后,就有可能找到解决方案。



8.21 交叉与匹配操作

8.7.2 CAP 软件中的背包加密法

令 Bob 和 Alice 满意的是,CAP 软件实现了背包加密法。同样,令 Eve 也高兴的是,CAP 也提供了破解背包加密法的 GA 工具。要运行 CAP 的背包加密法,选取 Cipher 菜单下的 Knapsack 项,弹出 Knapsack 窗口。CAP 可以帮助构建大背包,在 Knapsack 窗口中,选取 Key 菜单下的 Create 项即可,如图 8.22 所示。

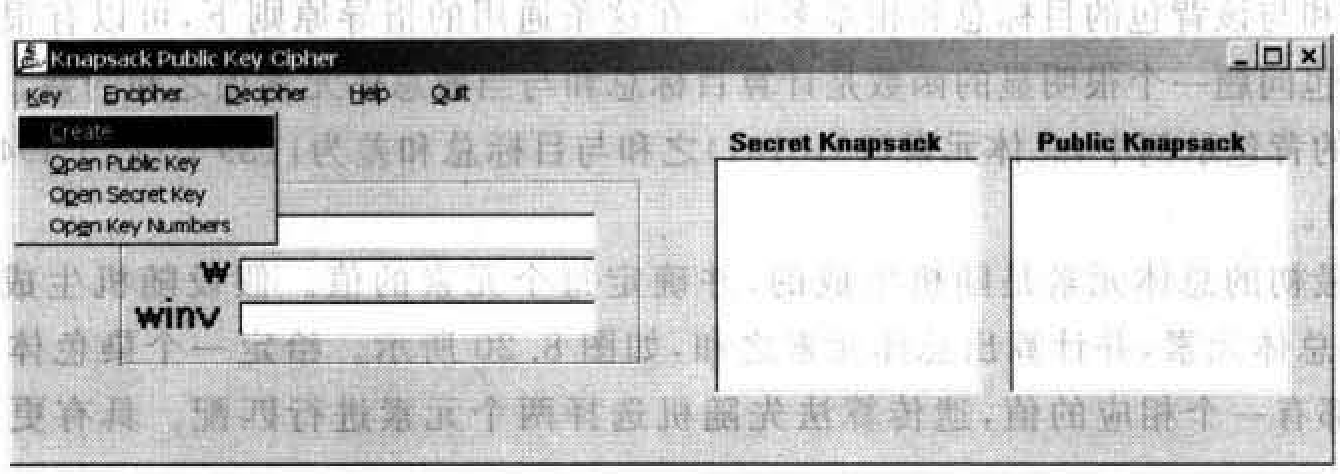


图 8.22 Knapsack 窗口

如图 8.23 所示,Bob 使用 CAP 软件生成了一个背包加密法密钥。他想生成一个含有 50 个元素的背包,因此他指定第一个元素为 23。为了在超增加背包中生成随机度,Bob 在 Max Difference 框中输入 50。由于超增加背包的惟一要求是,每个元素比前面所有元素的总和更大,Max Difference 框的目的是指定这种差值的最大值。

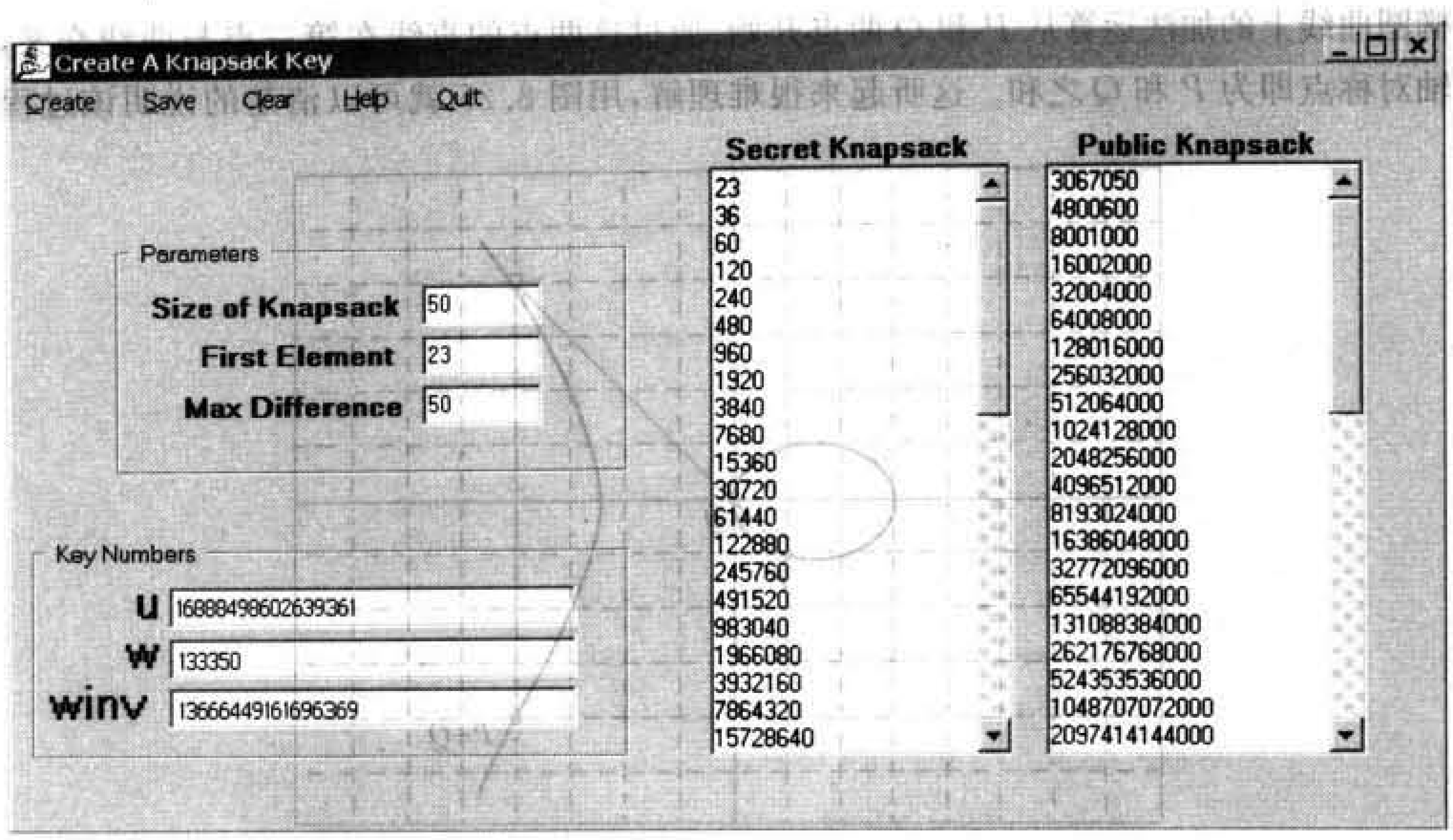


图 8.23 用 CAP 软件生成 Knapsack 密钥

8.8 椭圆曲线加密法

1985 年，Koblitz 和 Miller 在各自开发的新的公钥加密法中提出了使用椭圆曲线。椭圆曲线是由如下的方程式定义的：

$$y^2 = x^3 + ax + b \pmod{p}$$

其中 p 为一个素数或 2 的幂(有时也称为椭圆曲线的秩)，且 $4a^3 + 27b^2 \neq 0$。典型的椭圆曲线如图 8.24 所示。在椭圆曲线上，令我们感兴趣的是那些坐标值为整数的点。曲线上坐标值为整数的点的数量是有限的，这些点的数量大小对椭圆曲线加密法的安全性起着很大的作用。事实上，整数点可以组合成加法运算的形式，这是一种特别为椭圆曲线定义的操作。

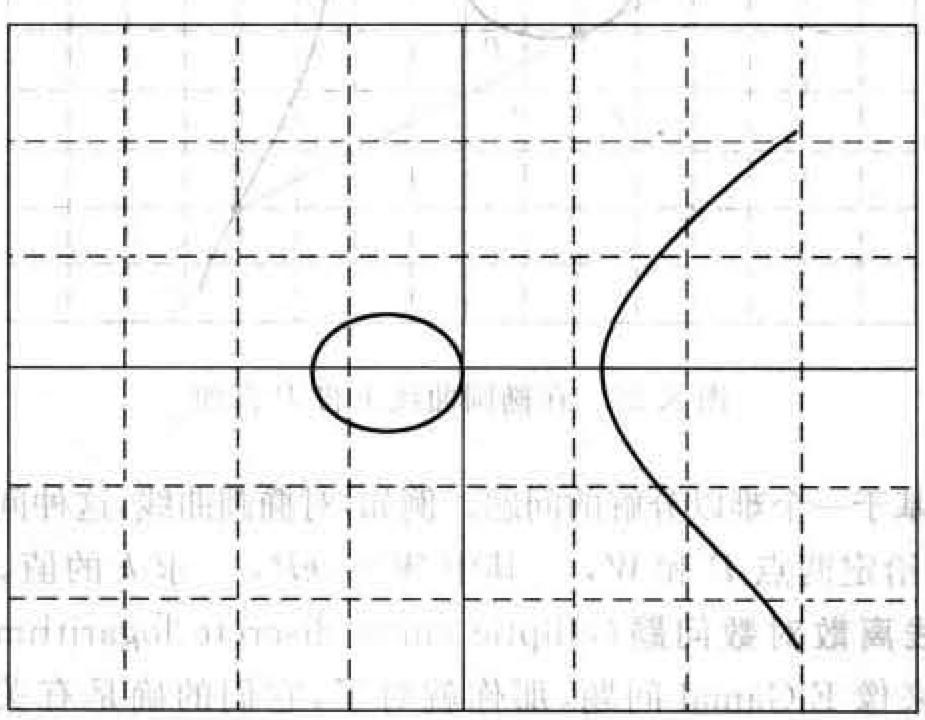


图 8.24 典型的椭圆曲线

椭圆曲线上的加法运算从 P 和 Q 两点开始，通过这两点的直线在第三点与曲线交叉，该点的 x 轴对称点即为 P 和 Q 之和。这听起来很难理解，用图 8.25 就可以清楚的说明该过程了。

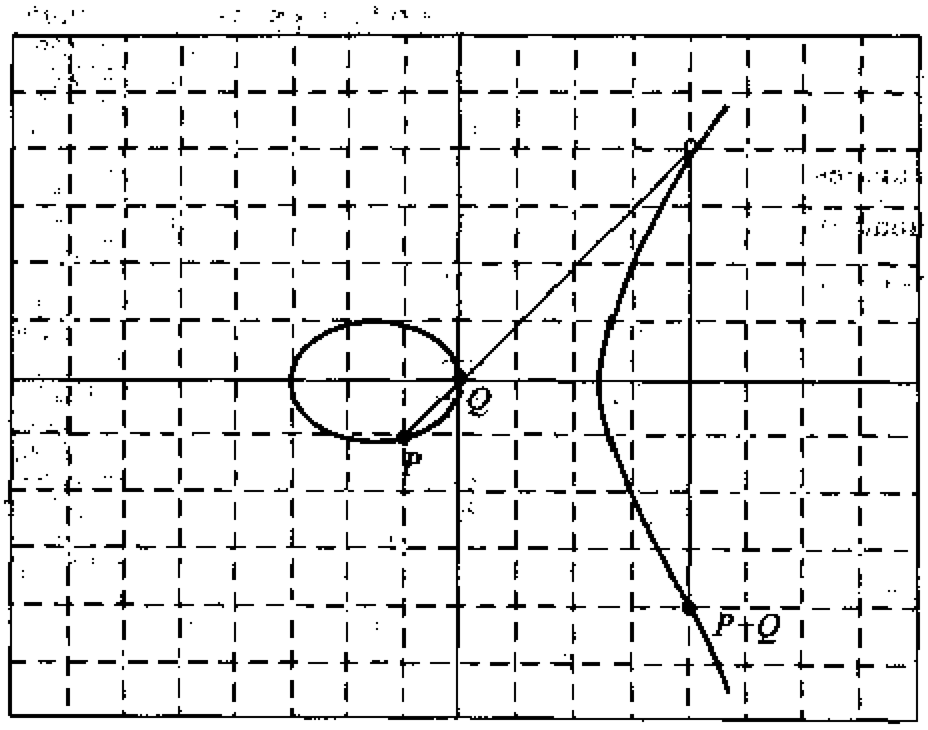


图 8.25 椭圆曲线的加密运算

对密码学家来说，对椭圆曲线加法运算真正感兴趣的是一个点与其自身相加的过程。也就是说，给定点 P，找出 $P+P$（即 $2P$）。如图 8.26 所示，画过 P 点的切线，该切线与曲线相交于一点，该点在曲线上的对称点即为 $2P$。P 点还可以自加 k 次，得到一点 W，且 $W=kP$。例如，如图 8.27 所示为 $3P$ 点。

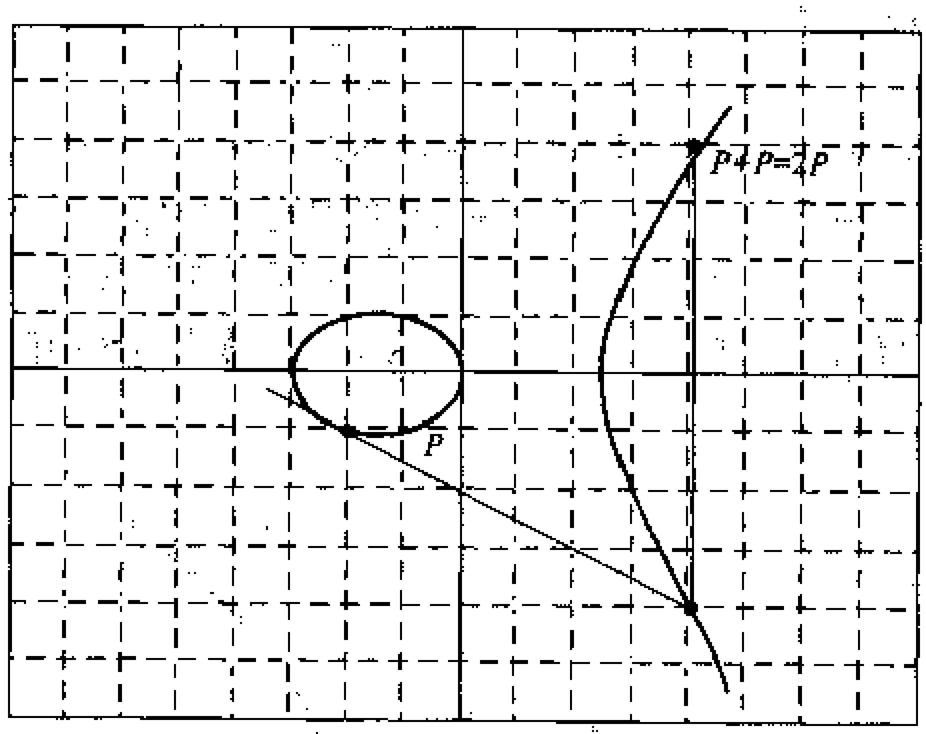


图 8.26 在椭圆曲线上将 P 自加

公钥加密法必须基于一个难以分解的问题。例如，对椭圆曲线，这种问题的描述如下：

给定两点 P 和 W， 其中 $W=kP$， 求 k 的值。

这称为**椭圆曲线离散对数问题**（elliptic curve discrete logarithm problem，ECDLP）。（如果你认为这听起来像 ElGamal 问题，那你就对了，它们的确是有关联的。）ECDLP 可以用穷尽查找法来求解。如图 8.28 所示，已知条件为点 P 和 Q，将 P 不断自加，直到到达 Q 点。对于一个大的 k，这不会比分解 k 简单。

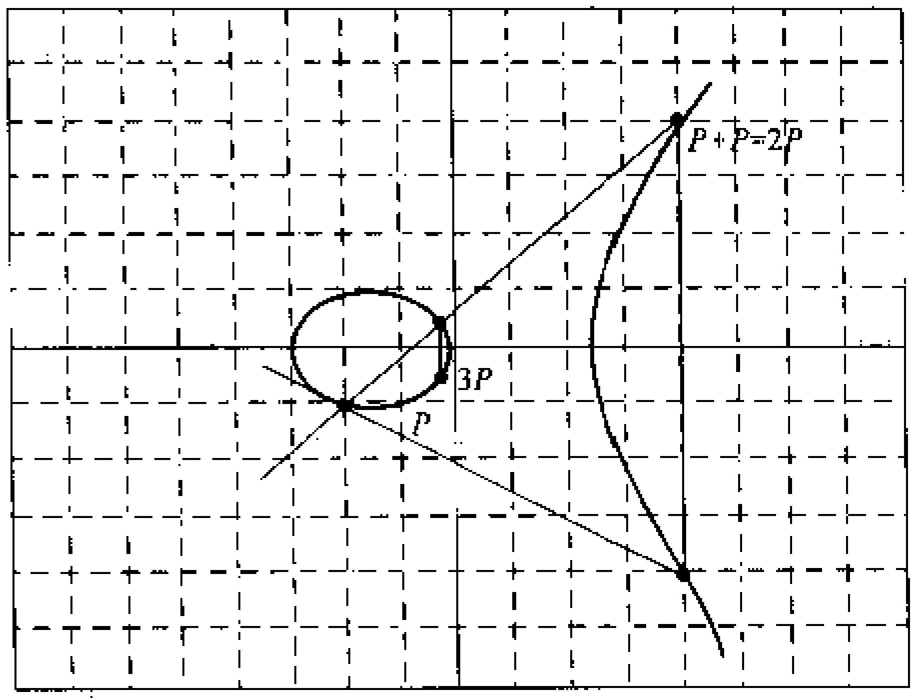


图 8.27　查找椭圆曲线上的 3P 点

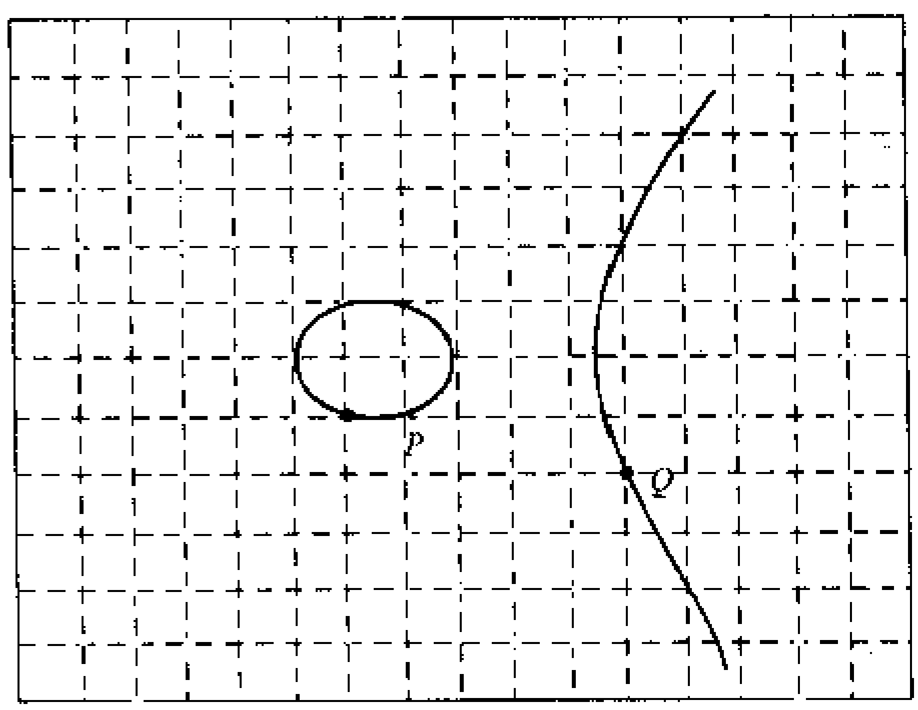


图 8.28　Q 是 P 的多少倍?

将 ECDLP 嵌入到加密系统的方法有多种。Bob 和 Alice 决定使用最简单的一种实现方法。Alice 首先选择一条椭圆曲线以及该曲线上的一点 G,然后选取一个保密数字 k,该数字将作为她的私钥。她的公钥为 G 和 P_A,其中 $P_A=kG$,并将公钥给 Bob。但 Bob 要给 Alice 发送一个消息时,他需要先将明文转换成数字 m,并在曲线上找出一点 P_m,使其 x 坐标值与 y 坐标值之差为 m。再选择一个随机数 r,然后用以下等式加密后将密文发送给 Alice:

$$C = \{rG, P_m + rP_A\}$$

通过用私钥与第一点相乘,并减去第二点,Alice 就可以很容易地解密:

$$P_m + rP_A - k(rG) = P_m + r(kG) - k(rG) = P_m$$

8.8.1　算法评述

尽管求解 EDCLP 的算法有很多种,但最快的要算 Pollard Rho 方法。然而,如果密钥很大,即使是 Pollard Rho 方法也要花费很长的计算时间。2000 年,为了将 108 位的 ECC

密钥破解，利用 Pollard Rho 方法，在 9500 台计算机上运行了 5 个月，这相当于在单台 PC 机上运行了 500 年的计算时间。要实现同等级别的 RSA 需要有 600 位的密钥。总而言之，ECC 使用较小的密钥就可以提供比 RSA 更高的安全级别。

下面列出了实现同等级别的安全性所需要的密钥大小，从中可以看出，ECC 比 RSA 更好：

安全性	RSA	ECC
低	512	112
中	1024	161
高	3072	256
很高	15360	512

正像 DES 有某些易破解的密钥一样，ECC 也有一些破解的曲线。例如，1993 年 Menezes、Okamoto 和 Vanstone 发现，有一类超简单(supersingular)曲线可用于指数攻击法来破解。这种方法称为 MOV 攻击法(以那篇论文的作者的名字命名的)因此，可用作加密法的曲线必定不能属于超简单的类型；这意味着，要使曲线能用于 ECC，必须满足以下条件：

(1) 如果 N 为整数坐标值的个数，那么它必须能被某些大素数整除，即 $N=kr$。

(2) 如果曲线有秩 p(即进行 mod p 运算)，那么当 i 属于小数字集合(即 $i=0\sim20$)时，r 不能整除 p^i-1。

另一类易破解的 ECC 曲线称为不规则曲线。这些曲线的整数坐标值的个数等于其秩(使用上面的定义就是 $N=p$)。事实上，为了避免使用易破解的曲线，NIST 在 2000 年推荐使用以下 10 种常用椭圆曲线，其中 5 种为素数秩，5 种为二进制秩：

素数秩 $p192=2192-264-1$

$p224=2224-296+1$

$p256=2256-2224+2191+296-1$

$p384=2384-2128-296+232-1$

$p521=2521-1$

二进制秩 $p=2163$

$p=2233$

$p=2283$

$p=2409$

$p=2571$

8.9 公钥加密法的应用

公钥算法最常见的应用包括保护文档的完整性，它提供了与手写签名同等的安全级别。因此，关于公钥加密法的全面介绍将在第 9 章中进行(在第 9 章中介绍了数字签名的概念)。公钥算法另一个常见的应用是保护对称块加密法密钥的完整性。

与块加密法相比，公钥算法的问题是运算速度非常慢。因此，用公钥加密系统加密一个

大文档效率不高，尤其是如果能用块加密法加密时，其速度是公钥加密法的 10～100 倍。而块加密法的问题是，要在两个用户之间安全传递密钥很困难。这正好对公钥算法来说就不是问题了，因为它的加密密钥总是公开的。解决这两个问题的方法是同时使用这两种加密法，这样，每种加密法都是最有效地使用。例如，如果 Bob 想尽快地给 Alice 发送一个大文档，他就可以使用 AES 来加密，然后发送给 Alice。当然，Alice 需要 Bob 用来加密的 AES 密钥。如果 Bob 直接将密钥发送给 Alice，那么他就相当于没有加密一样，因为 Eve 可以截获该密钥和信息。因此，Bob 使用 Alice 的 RSA 公钥来加密他的 AES 密钥。Alice 使用她的 RSA 私钥还原实际的 AES 密钥，然后用该密钥解密加密文档，从而就可以阅读文档了。尽管 RSA 运算慢，当它只是用来加密解密相对较小的 AES 密钥。同样，尽管 AES 密钥很难记住，但 Bob 并不需要记住它，只需要将它作为一条要加密的消息，使用 Alice 的公钥加密后发送给 Alice 即可，从而可以确保安全。

这里应该注意的问题是 Bob 与 Alice 之间的密钥管理。这个问题的重要性不是第一次出现了。密钥管理与加密算法选用具有同等的重要，这将在下一章中详细介绍。

8.10　公钥加密法的历史简介

人们通常认为公钥系统是由 Merkel、Diffie 和 Hellman 于 1976 年首先提出的，当时他们描述了背包算法。两年后，Ronald Rivest、Adi Shamir 和 Leonard Adelman 发表了一篇论文，介绍了 RSA 公钥系统如何出现的。但是，直到 1997 年，人们才最终知道公钥加密法的秘密历史。不是说 Merkel、Diffie、Hellman、Rivest、Shamir 和 Adelman 的工作不重要（毕竟他们各自独立地发现了公钥加密法的基本概念），但的确是 James Ellis 于 1970 年发现了公钥加密法，当时，他供职于英国政府通信总部（Bletchley Park 的前身）的通信电子安全组。他的工作直到 1997 年才发表，因为英国政府一直没有解禁。1973 年，Clifford Cocks（当时也是供职于通信电子安全组）发表了一篇机密论文，描述了 RSA 的加密原理。

这说明了密码学领域的一个问题。实际上，有两个并行进行的力量在进行密码学的研究工作，一个是机密工作，出于保密状态；另一个是学术工作，可以公开发表。

8.11　本章小结

Bob、Alice 和 Eve 又学习了一种新的加密法——公钥加密法，这种加密法需要有两个密钥，一个为公钥，另一个为私钥。这不只是公钥加密法的惟一特性。与其他加密法依靠复杂的替换和置换过程来实现安全性不同，公钥加密法是依靠求解某些数学问题的难度来保证安全的。

Bob、Alice 和 Eve 还发现，数学知识在新加密法的开发、分析和评估中起着很重要的作用。他们尤其对数字和数论知识有了进一步的了解。

Bob 和 Alice 知道，对于公钥加密系统，他们需要生成的复杂密钥应是非常大的数字。这又引入了一些新的问题：他们如何管理这些密钥，当他们需要时可以使用它们，但 Eve 却

不能？要解决这个问题，他们需要学习下一章的内容。

Eve明白了，要破解这种新的加密法，不仅需要有耐心，还需要花些时间在数学知识的学习上。因为公钥加密系统的安全性是与某些数学问题求解的困难度相关的，因此，如果Eve能找到一种好的办法，充分利用计算机的速度来解决目标问题，她就可以破解Bob与Alice之间的通信信息了。如果不能对公钥算法进行直接攻击，Eve还可以通过其他途径寻找信息来破解它。

最后，我们的加密法结构图又添加了一种新的加密法，如图8.29所示。

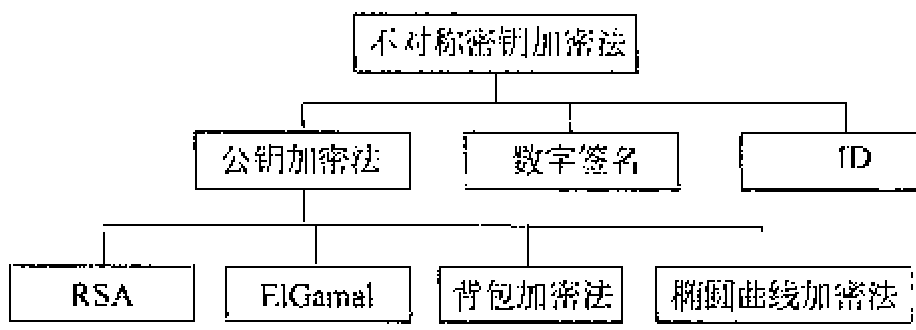


图8.29 公钥加密法的分类

8.12 重要术语

ElGamal加密法	一种公钥加密法，其安全性是基于求解离散对数问题的困难度的。
椭圆曲线	$y^2=x^3+ax+b \pmod p$形式的方程，其曲线上的整数坐标点加法运算可以用来创建一种加密系统。
欧几里得	用来寻找两个数的最大公约数的方法，其推广方法可以用来计算取模的逆运算。
基因算法	一种随机查找算法，基于生物学的自然选择特征。
背包加密法	一种基于求解子集和问题的困难度的加密法。
公钥加密系统	一种具有两个密钥的加密系统，公钥用于将明文加密，私钥用于解密。
RSA	一种广泛使用的公钥加密系统，其安全性是基于将大整数因子分解的困难度的。
重复平方法	一种用来减少RSA（和其他使用乘法运算的加密算法）所需的乘法次数的方法。
旁道密码分析	通过监视加密法实现的特征（如功率损耗或运行速度），来破解加密法的一种方法。

习　题

1. Bob想使用RSA（块大小为10）将以下消息加密并发送给Alice：

Hi Alice it has been a while since I contacted you. I want you to know that everything is going fine and I will call you tomorrow. Bob

Alice 的公钥如下：$n=96498667$，$e=94475891$。请将以上消息生成相应的密文。

2. 假设从问题 1 知道了 Alice 的公钥，请找出她的私钥，并用该私钥将 Bob 发送给 Alice 的密文解密。

3. 以下数字哪些是素数（请说明你所来的测试方法）？

(a) 12253

(b) 900213345

(c) 17515889

(d) 161270444129

(e) 453851

(f) 7670175557761

4. 以下数字都是素数：

453851　1207039　32046463　398163641　1845277897　40847631499

请用图形表示出验证以上数字所需时间与其位数的关系。使用两种尾数除法可以在图中画出两条线。从中你能得出什么结论？

5. 以下数字都是素数：

651247415327　8301310004401　18198400004401　449063710848593

5264651305119901　68063301125552099　171382356359199751

4913778538696132955551　85609598896830600941057

请用图形表示出验证以上数字所需时间与其位数的关系。分别用 Fermat、Lehmann 和 Miller 方法进行验证，可以在图中画出 3 条线。从中你能得出什么结论？

6. Eve 截获了从 Alice 发送给 Bob 的消息，该消息使用的加密法是 RSA（块大小为 16）。她知道 Bob 的公钥为：$n=959455486236$，$e=27923157781$。

402570513855

9110026864

153439736421

864921693668

637925583027

909658537600

878879266527

728359495475

593068589632

916200064152

19746950663

264468802305

362546308473

46544990900

595811115816

793018157000

237037688464

38552711197

532693437264

560347854564

475636675945

822754706889

270893972249

请找出其明文。

7. 将以下数字因子分解(请说明所使用的方法):

(a) 15961

(b) 76001

(c) 175228969

(d) 323235792283

(e) 468603770014481

8. 将以下数字因子分解,并请用图形表示出因子分解所需时间与其位数的关系。请尽量使用尾数除法(利用 CAP 软件),并分别使用 Factoring 方法(位于 Analysis 工具下的 Public Key Systems 中)下的 4 种方法,这样就可以得到 5 条曲线。这些数字是:

(a) 503219

(b) 1200559

(c) 79811027

(d) 545266873

(e) 154306682903

(f) 4775027828531

(g) 526145434732711

(h) 224192040175009871

9. Bob 使用 ElGamal 加密法和 Alice 的公钥将以下消息加密后发送给 Alice,Alice 的公钥为 $p=11851409$,$g=27046$,$b=1848340$:

Alice,It took a while for me to generate my ElGamal key. Is that normal? Bob

请生成 Bob 发送给 Alice 的密文。

10. 用图形表示出素数大小与 ElGamal 加密法生成 p 和 g 所花费的时间关系,其中素数大小为 6~15 位,素数因子个数为 2。从中你可以得出什么结论?

11. Alice 使用有 15 个元素的背包加密法将以下消息加密并发送给 Bob:

Bob,these knapsacks are getting out of hand. What should we do? Alice

Bob 的公钥为:

301

387

774

1548

3096

6129

12384

24768

49536

99072

50686

101372

55286

110572

73686

请生成 Alice 发送给 Bob 的密文。

12. 使用快速指数法计算 $123^{79} \bmod 77$ 的值，需要进行多少次乘法运算？这比直接计算节省了多少次？

13. 试找出一种将任意大小的数因子分解耗时不超过 10 秒钟的方法。

第 9 章　密钥管理、数字签名、散列函数与证书

9.0　概　　述

当 Alice 与 Bob 开始相互发送消息时，他们惟一关心的问题是 Eve 不能够读取消息内容。他们开始使用了几种经典加密法，但发现 Eve 能够破解这些加密法。他们转移到更复杂的现代加密法上，并且发现，如果他们小心仔细的话，能够防止 Eve 阅读他们的消息。但是，出现了一组全新的问题。其中第一个问题是管理新加密法所要求的大量随机密钥的问题。他们必须使用长达 56～256 位长度的密钥，并且，即使使用十六进制表示（四个位使用一个符号），这些密钥也很长，并且没有助记意义。（记一记 4ef0 124a 9732 cb44 5b01 6821 cc07 397f 试一试，这是一个 128 位的密钥）。因此就产生了这样的结果：Alice 与 Bob 倾向于在某个东西上写下密钥，并把它保存在他们的计算机的附近（比如放在键盘下面——谁会去这里找东西呢？）很明显，管理密钥已经成为了一个问题。

甚至更危险的问题也开始出现了。某天，Bob 收到了一封来自 Alice 的消息，请求 Bob 向她发送一些重要文件。消息使用 Bob 的公钥加密，并告诉他 Alice 有了一个新的公钥，他应该使用这个新公钥加密发送给她的数据。Bob 使用新的公钥，它包含在 Alice 的消息中，并发送了文件。一周后，当 Bob 询问 Alice 这些文件时，Alice 说她对此一无所知。更糟糕的是，Alice 说她从来没有更换过公钥。

后来，Alice 收到了一条来自 Bob 的加密消息，授权使用公司资金从特定公司购买一件高价的东西。Alice 对 Bob 的行为感到困惑，打电话确认 Bob 所发送的消息。当然，她不想在电话中讨论消息的内容，否则的话，他又何必以加密方式向她发送消息呢？Bob 确认他确实向她发送过消息，因此，Alice 就购买了那个东西。第二天，Bob 打电话问 Alice 她为什么要购买那个昂贵的东西，而他的消息只是请她购买要便宜得多的东西。现在似乎 Alice 成了牺牲品。从现在开始，无论是 Bob 还是 Alice 都不再相信他们之间相互发送的消息，即使这些消息使用高强度算法加密时也是如此。情况似乎是这样：仅仅加密文件和消息还是不够的——在事情有转机之前，他们需要恢复某种程度的信任。

他们已经发现，即使 Eve 不能够阅读他们的消息，依然有可能截获消息并发送一个修改后的消息版本，这是一个艰难的困局。Bob 如何才能够确保 Alice 是真正的作者呢？还有，如果 Alice 在她的消息中做出了某种承诺，她可以在以后否认这些承诺吗——声称是 Eve 冒用她的名义做出的承诺？忽然之间，密码学成为了一种远远不止是隐藏信息的科学；使用密码学找到确保消息认证和共享密钥的途径也是十分重要的。现在到了回顾一下前面的章节并探索密码学新应用的时候了。

回顾一下第 1 章，密码学的主要用途是提供消息的机密性。它确保只有得到授权的用户才能够真正地阅读消息。现在，还需要一层完整性，用于确保在解密的时候消息没有以任何方式被修改过。除此之外，还需要一个认证层，以便确保消息作者的真实性。消息应该来

自声明发送了消息的人，消息的作者以后也不能否认曾经发送的消息。

本章将向你展示如何在消息中添加完整层和认证层，以及如何有效和安全地管理密钥。这些内容将应对四种类型的攻击：

- **操纵**(Manipulation)——截获并修改消息
- **伪装**(Masquerade)——假装为另一个用户
- **重放**(Replay)——记录并(在以后的某个日子)重放通信
- **否认**(Repudiation)——参与通信的一方或所有方否认通信

9.1 密钥交换

还记得德国人为每一次 Enigma 传输设置一个惟一密钥的原因吗？他们想避免一而再地使用同一个密钥。理论上讲，这是一个好策略；但是，在现实中，这又造就了盟军能够利用的另一个弱点。他们的观念是正确的——不重复使用密钥——但他们在为每次传输建立惟一共同密钥时采用了错误的过程。这现在依然是密码学上的一个问题——如何在人们之间安全地共享新的密钥？

对于 Bob 和 Alice 来说，首要任务是拿出一种管理他们所需的大量密钥的方法，并且这种方法不受 Eve 的干扰。虽然公钥能够完成这个任务，但公钥算法速度慢。因此，Bob 和 Alice 依然想使用速度更快一些的块加密法，这种加密法要求有一个只有他们两个人才知道的密钥。安全上还要求规范的密钥交换。因此，Bob 和 Alice 需要一种能够生成单个密钥的安全方法，这个密钥只有他们两个人知道，并且不需要这两个人见面即可完成密钥交换。

现在已经有几种密钥交换算法可以使用——其中绝大多数算法都基于公钥系统。最先开发的算法之一称为 Diffie-Hellman 密钥交换系统。要建立 Diffie-Hellman(DH)协议，Bob 和 Alice 必须首先共享两个数：p 为一个大素数；g 为一个小于 p 的随机数。由于 p 和 g 是公开的，因此 Bob 和 Alice 可以在一个不安全的通道上选择这两个数(在这一点上，他们也可以把这两个数发送给 Eve)。Alice 选择一个保密的随机数 a，并将值 $g^a \bmod p$ 发送给 Bob。与此同时，Bob 选择一个他自己的保密随机数 b，并将值 $g^b \bmod p$ 发送给 Alice。Alice 使用自己的保密随机数和 Bob 发送给她的值计算下式：

$$(g^b \bmod p)^a \bmod p = k$$

而 Bob 使用他自己的保密随机数和 Alice 发送给他的值来计算下式：

$$(g^a \bmod p)^b \bmod p = k$$

他们最终得到了一个相同的数 k，这个数成为了他们的共同密钥。

从这个过程中我们可以观察到下述几点：

(1) Bob 和 Alice 都不关心最终的密钥到底是什么。

(2) Bob 和 Alice 相互之间都不分享他们各自的保密数。

(3) Eve 能够得到 g、p 以及值 $g^a \bmod p$ 和 $g^b \bmod p$，而 Eve 得到 k 的惟一方法是计算出 a 和 b，这个问题等价于离散对数问题。

例如，如果 Alice 和 Bob 同意使用值 $p=7682567015711 5949022871$ 和 $g=129633$，之后，Alice 选择她的保密值为 1567，Bob 选择他的保密值为 3001；他们分别计

算要相互发送的数。Alice 得到她要发送给 Bob 的值为 374886062511822177 6574，Bob 发送值 3412692391758026181 3405 给 Alice。最终得到了下面的共同密钥：

$$323478797153776277 49156$$

现在，这个数字只有 Bob 和 Alice 知道，它可以以几种方式用作密钥来加密消息。例如，他们可以把这个数值转换为十六进制数(Hex)，并把前八个十六进制数用作公用的 DES 密钥。他们也可以把这个数用于流密钥，其中部分数字用作反馈循环，部分数字用作 LFSR 的初始设置。这个密钥的十六进制表示为：

DB32A057D72A6FEB648

如果他们使用 DES 的话，可以选用高八位数字作为密钥，这里就是 DB32A057。

Bob 和 Alice 很快就意识到，这些数字太大了，难以手工使用。幸运的是，他们发现，CAP 软件可以帮助他们完成所有的事情，包括将公用密钥转换为十六进制格式，如图 9.1 所示。

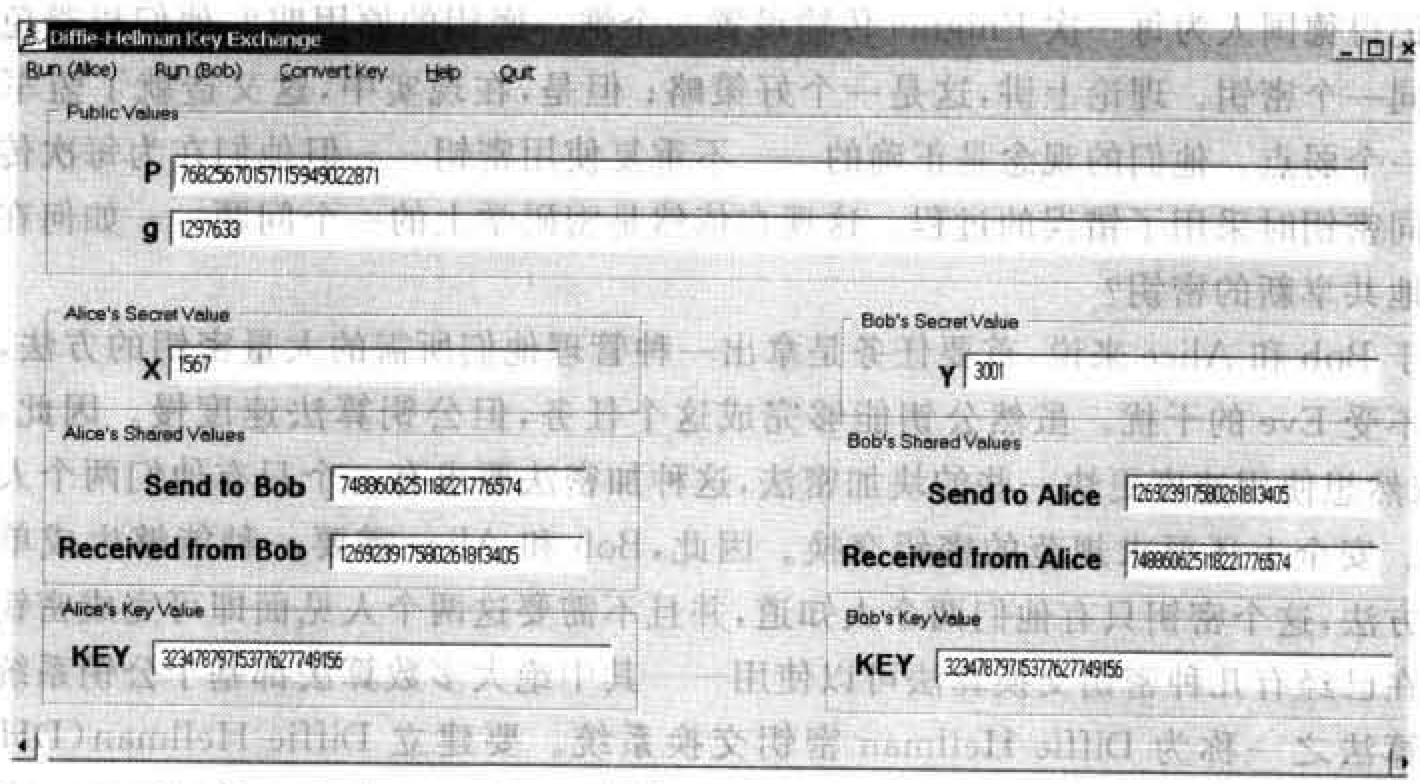


图 9.1　Diffie-Hellman 密钥计算示例

全面的 DH 过程并不试图使用指数。其实也可以使用 Diffie-Hellman 方案的椭圆曲线加密法(ECC)版本。此时，Alice 和 Bob 选择一个适宜的曲线 G 和基点 P。每个人都选择一个保密的随机数(Alice 选 A，Bob 选 B)。Alice 将值 $P_A = AP$ 发送给 Bob，Bob 将值 $P_B = BP$ 发送给 Alice，他们每一个人都是用发送给自己的值和自己拥有的保密值计算公用的 ECC 密钥：$K = ABP$。

只有当公开的数字足够大时，Diffie-Hellman 方案才是安全的。即使这些数选择得很大，但 Diffie-Hellman 方案依然会遭到中间人(man-in-the-middle)攻击。问题在于，由于 Alice 和 Bob 只能在一个不安全的通道上通信，那么，他们怎么能够知道通信的另一方到底是谁呢？Alice 如何能够知道 Bob 就是发送给她值 $g^b \bmod p$ 的那个人呢？如果 Eve 能够截获 Bob 和 Alice 之间的消息，它就能够建立两次密钥交换(一次同 Alice，另一次同 Bob)让 Alice 和 Bob 都觉得他们得到了共同密钥。这种攻击方式如图 9.2 所示。

Diffie-Hellman 的认证版本也可以用于防止中间人攻击。这个版本要求 Bob 和 Alice 都有用于帮助建立 Diffie-Hellman 过程的保密和公钥。像前面所讲的那样，Bob 和 Alice 公开地分享素数 p 和数字 g。Alice 还有一个大的保密密钥 A 和公钥 g^A。Bob 有一个大的保

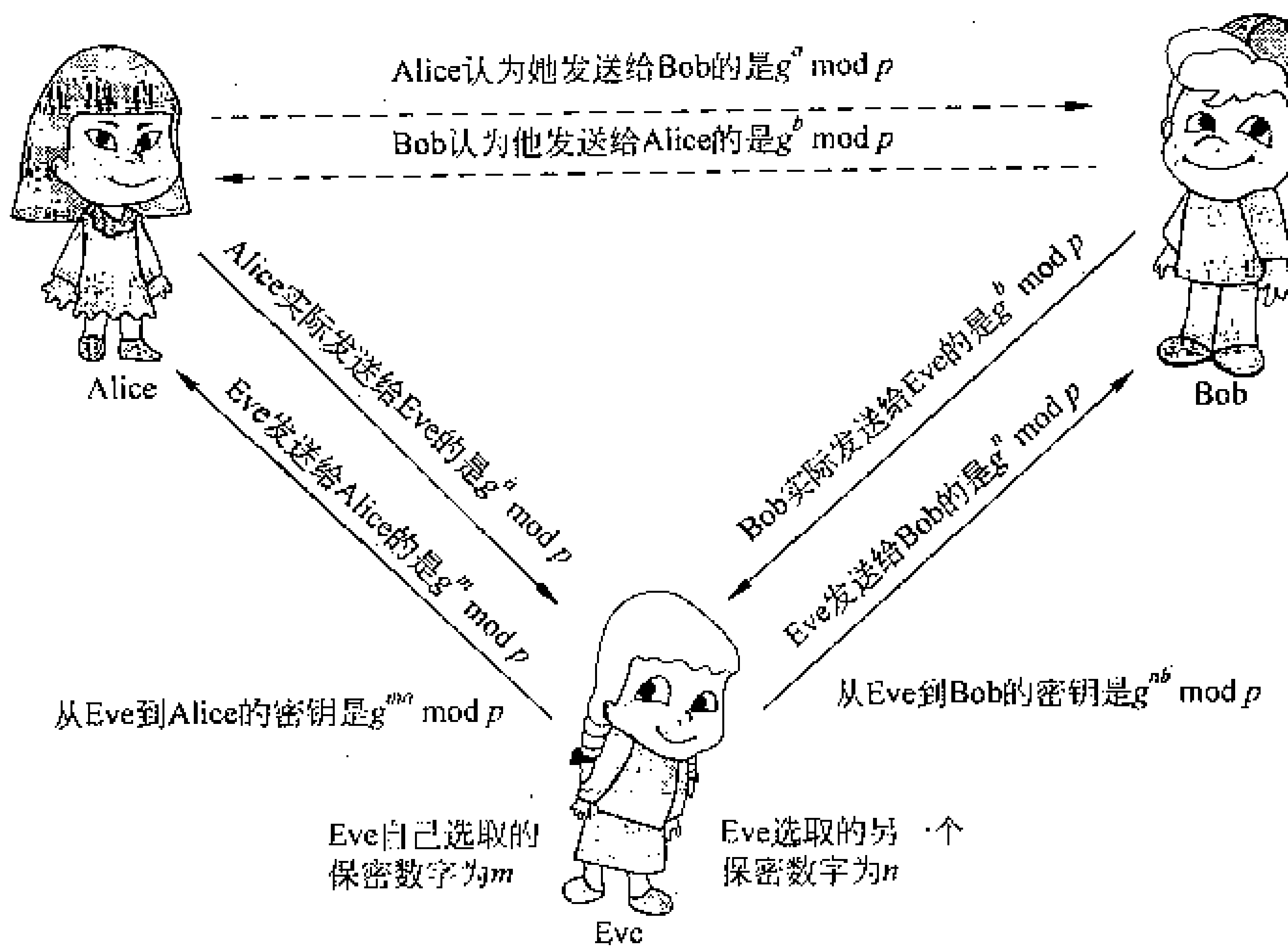


图 9.2　中间人对 Diffie-Hellman 的攻击

密密钥 B 和公钥 g^B。他们两个人都可以使用对方的公钥和自己的私钥来确定 $K=g^{AB}$。Alice 计算 $(g^B)^A$，而 Bob 计算 $(g^A)^B$。由于 Eve 既不知道 A、也不知道 B，因此她不能够确定 K；这样，K 就可以用作认证值。DH 过程变成了：

(1) Alice 选择一个随机数 a。

(2) 她向 Bob 发送 g^{aK}。

(3) Bob 选择一个随机数 b。

(4) 他向 Alice 发送 g^{bK}。

(5) Bob 按下述方式计算共同密钥：

$$K_{AB} = (g^{aK})^{K^{-1}b} = g^{ab}$$

Alice 得到相同的密钥值：

$$K_{AB} = (g^{bK})^{K^{-1}a} = g^{ab}$$

你或许在无意识的情况下已经使用了 Diffie-Hellman 过程。它是用于保护 Web 事务安全的安全套接层(SSL)中的一个常用选项之一。SSL 的任务是在你和你所选择的 Web 网站之间建立一条保密链路，而不需要输入加密密钥。其结果是，你的计算机和 Web 服务器使用 Diffie-Hellman 为当前的 Web 会话建立一个公用密钥。每次你要访问一个安全的 Web 站点时，都自动建立一个新的密钥。

9.1.1　Internet 密钥交换过程

像在前一节说明的那样，Diffie-Hellman 过程用于为安全的 Internet 连接建立一个公用密钥。回顾一下第 7 章(准确地说，是 7.6.2 节)，那里简要地讨论了建立安全 Internet 连接的过程。并告诉你等到第 9 章再了解如何为安全链路构造密钥。好了，你的等待时间结束了。

IPSec 协议中这类链接的正式过程称为 Internet 密钥交换(IKE),它用于建立安全关联(SA)。例如,如果 Alice 和 Bob 想要一个安全的 Internet 连接,IKE 进程将在机器之间发起并交换六条消息。请注意,Alice 和 Bob 并没有真正的构造或发送这些消息;而他们是 IKE 自动过程的组成部分。图 9.3 显示了这六条消息。通过发送 IKE SA 建议,Alice 的机器启动连接过程。这个建议中列出了她的计算机能够实现的加密算法的类型以及安全链路所需的其他关键信息。Bob 的机器通过从 Alice 的列表中选择它能够使用的加密算法与其他特性来进行响应。如果两台机器之间存在一组相同的方法,那么协议继续进行。下两条消息跟随着 Diffie-Hellman 过程。Alice 的机器发送她那一部分的 Diffie-Hellman 密钥和随机数值,Bob 的机器使用他那一部分的 Diffie-Hellman 密钥和随机数值作为响应。现在,双方都拥有了共同的保密密钥。最后,Alice 的机器发送它的标识信息,Bob 的机器以他自己的标识信息作为响应。

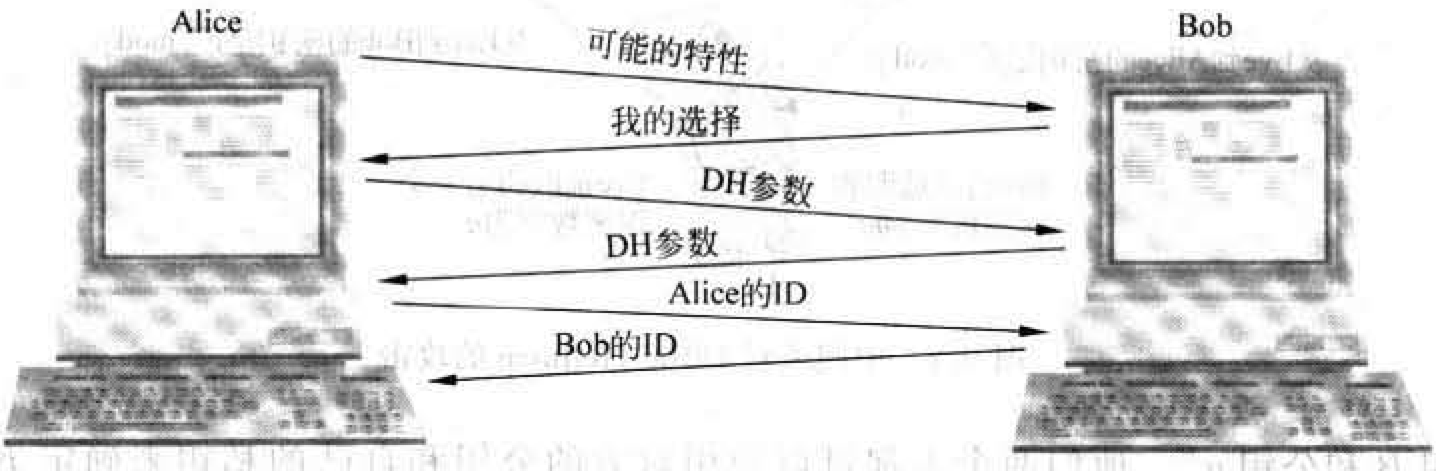


图 9.3 Internet 密钥交换过程

9.1.2 组密钥

对成对的用户来说,Diffie-Hellman 工作良好,但人们还需要为众多用户安全地建立一个已知的密钥。当有人想向几个用户广播一条消息时,这种需求进一步上升了。这种方式称为多播(或一对多)通信链路,我们在下一节进行介绍。另一些时候,当一组用户需要在一个多对多的通信链路上进行安全讨论时,也进一步提升了这种需要。此时,每一个发送者也同时都是接收者,这个过程称为对等组通信(peer-group communication)。

可以通用化 DH,以便能够在对等组模式工作。例如,比如 Chris 加入到 Bob 和 Alice 的通信中。他们需要一个共同的密钥,这样,他们选择使用 DH。所有的三个人都同意了共有值 g 和 n。Alice 开始选择一个随机数 a,并把值 $g^a \bmod n$ 发送给 Bob。Bob 选择他自己的随机数 b,并把发送给 Alice 的值 $g^b \bmod n$ 发送给 Chris。最后,Chris 选择一个随机数 c,并把值 $g^c \bmod n$ 发送给 Alice。这样就完成了第一步的工作。现在 Alice 向 Bob 发送值 $g^{ac} \bmod n$,Bob 向 Chris 发送值 $g^{ab} \bmod n$,Chris 向 Alice 发送值 $g^{bc} \bmod n$。这样完成了第二步工作。第三步也就是最后一步的工作是,每一个人都计算公共的密钥。Alice 使用她自己的随机数和 Chris 发送给她的值计算 $K=(g^{bc})^a=g^{abc} \bmod n$。Bob 使用他自己的随机数和 Alice 发送给他的最终值计算 $K=(g^{ac})^b=g^{abc} \bmod n$。Chris 使用他自己的随机数和 Bob 发送给他的最终值计算 $K=(g^{ab})^c=g^{abc} \bmod n$。他们都拥有了相同的密钥,但从来没有直接发送这个密钥,因此,这个密钥依然是安全的。现在,他们能够建立多播或对等组通信链路。图 9.4 演示了这个过程。

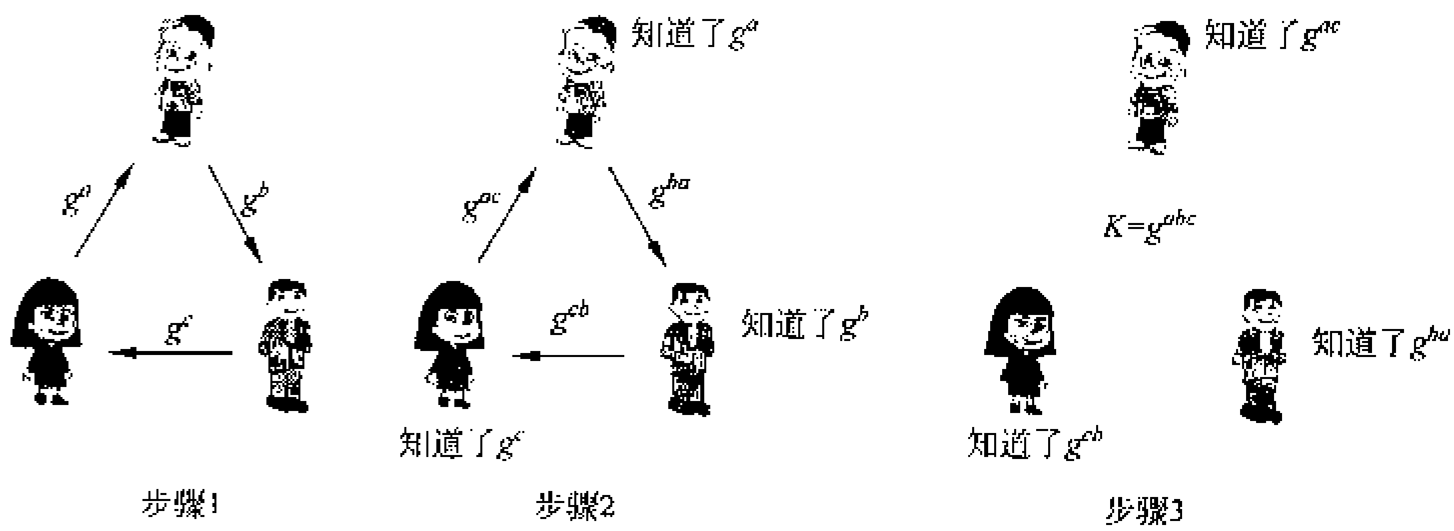


图 9.4　组 Diffie-Hellman 过程

虽然这种方法为 Bob 和 Alice 提供了标准 DH 过程的直接通用化方法，但它确实带来了一些问题。例如，某人想要加入到他们的组中，他们就不得不再重复一次这样的步骤。不仅如此，由于参与者为四个用户，他们不得不经历一个四个步骤的过程而不是三个步骤的过程。如果某个人离开这个组，那么也必须再一次启动密钥分配过程。总而言之，这种过程没有提供大量用户的组和可变用户数量的组所需的灵活性。

另一方面，显然也可以将组中的一个成员分配为组控制员（Group Controller，GC）。GC 与组中的每一个成员分享**密钥加密密钥**（KEK，Key Encryption Key）。组中的所有成员（M_1，M_2，…，M_n）分享单一的**流量加密密钥**（TEK，Traffic Encryption Key），该密钥用于相互之间发送加密的安全消息。这个模型直观上为如图 9.5 所示的星型结构。

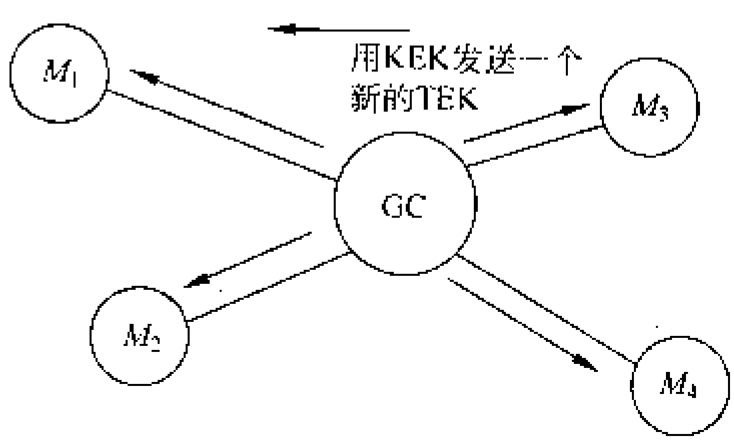


图 9.5　星型结构的组密钥管理系统

当新的成员加入该组时，GC 就为她建立 KEK，并确定新的 TEK。该 TEK 被发送给每一个成员，并使用各成员自己的 KEK 进行加密。如果某个成员离开这个组，GC 确定新的 TEK，并把它发送给组中留下的其他成员，当然，使用各成员自己的 KEK 密钥加密。这种方法的伸缩性并不好。一个有 n 个成员的组需要 n 个加密密钥和一个流量加密密钥，总共需要 $n+1$ 个密钥。

总的来说，良好的密钥管理系统不仅应该提供添加和删除成员的有效方法，而且也应该提供四种形式的安全性：

（1）组密钥安全性——组外的人应该不能够得到组密钥。

（2）后向安全性——从当前密钥的一个子集知识中不应该能够确定出前一个组密钥。

（3）前向安全型——从当前密钥的一个子集知识中不应该能够确定出任何未来要使用的密钥。

（4）密钥独立性——从当前密钥的一个子集知识中不应该能够确定出任何组密钥。

在 2000 年计算机协会（ACM）会议上，Kim、Perrig 和 Tsudik 展示了一种密钥管理模型，它提供了更好的伸缩能力和更正式的加入和退出组的过程。这个模型称为基于树的组 Diffie-Hellman 协议（TGDH）。密钥之间的关系被表示为密钥树，图 9.6 给出了一个 6 个成员的表示方式。图中每一个密钥方框包含了两个密钥：保密密钥 K_i 和公钥（称为盲密钥）BK_i。根密钥 K_0 为组密钥。每一个成员都能够访问所有的盲密钥和从根节点到成员自己节点路径上的所有保密密钥。例如，M_1 知道保密密钥 K_7、K_3、K_1 和 K_0，以及所有的公钥。

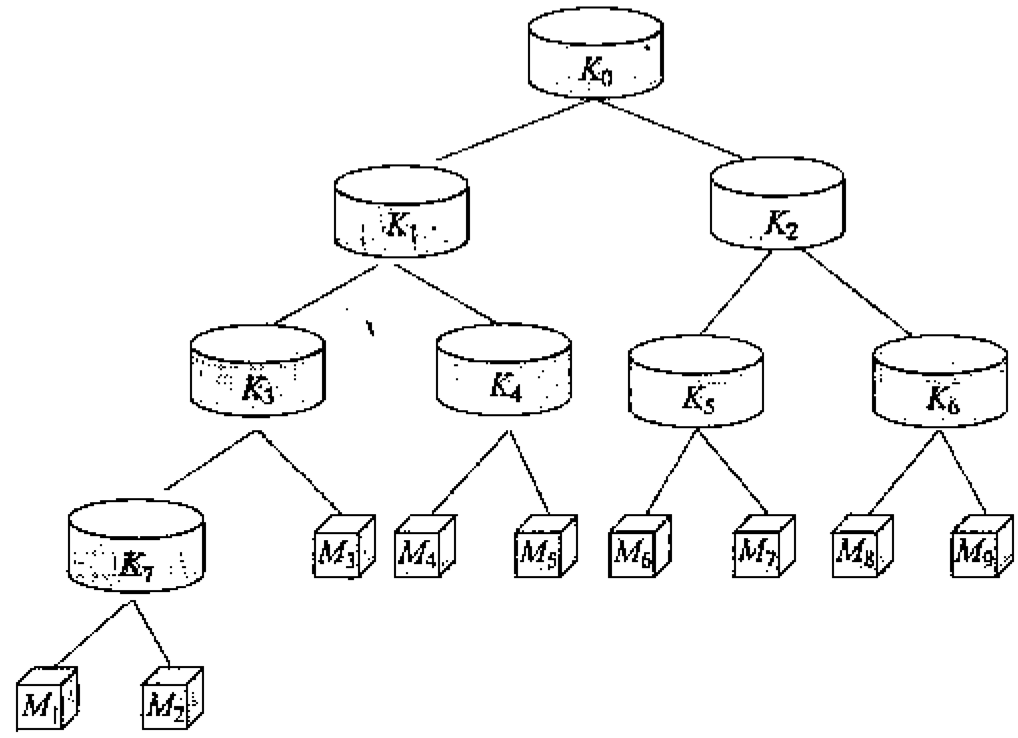


图 9.6　一个 TGDH 系统的示例

在开始创建这些密钥时，所有成员都像在 DH 协议中那样赞同公共值 g 和 p。之后，每一个成员选择一个保密的随机数（也就是说，成员 M_i 选择了 a_i）。如果两个节点都是树中同一个密钥节点的子节点（比如 M_1 和 M_2），那么他们遵从标准的 DH 协议，并基于各自的随机数创建一个共享的保密密钥。在示例中（所有值都省略了 mod p）：

M_1 向 M_2 发送值 g^{a_1}

M_2 向 M_1 发送值 g^{a_2}

使用这些值，他们计算出保密密钥值：

$$K_7 = g^{a_1 a_2}$$

和公开盲密钥值：

$$BK_7 = g^{K_7} = g^{g^{a_1 a_2}}$$

图 9.7 给出了示例中各阶段的密钥。每个节点的两个子节点使用 DH 协议为其父节点建立密钥。对于节点 K_3 来说，其过程应该是：

M_3 向 K_3 发送值 g^{a_3}

K_3 向 M_3 发送值 g^{K_7}

结果为：

$$K_3 = g^{a_3 K_7} \quad \text{并且} \quad BK_3 = g^{K_3}$$

由于每一个成员都知道从其位置上溯到根节点的保密密钥，这样每一个成员也就知道了组密钥。他们还具备了中间密钥的知识，这些知识可以用于构造安全子组。例如，M_1、

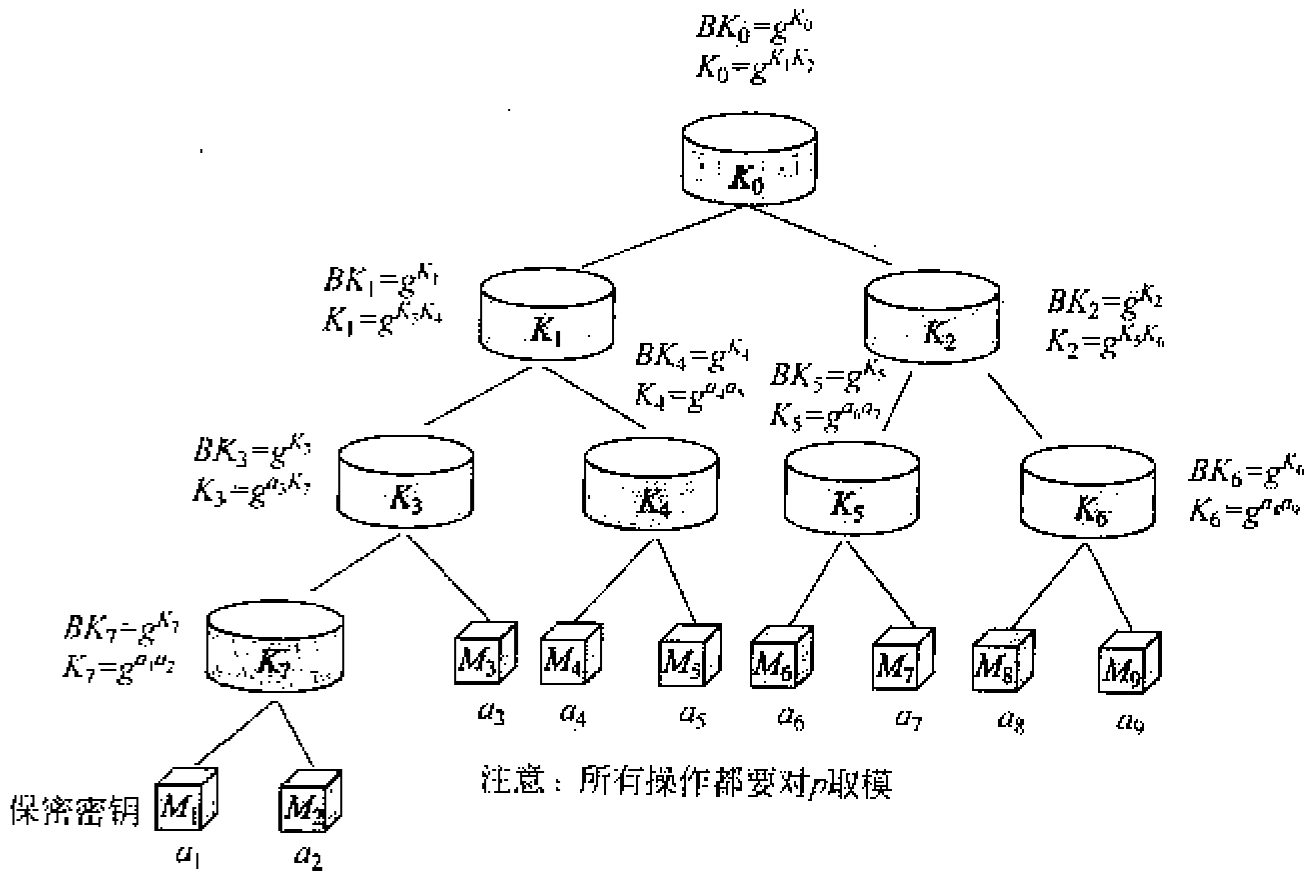


图 9.7　示例 TGDH 系统中的密钥

M_2、M_3 是惟一知道 K_3 的成员，他们可以把这个密钥用作他们之间安全通信的密钥。

如果某个成员离开了这个组，就必须改变某些密钥，以便老成员无法使用以前的密钥知识窃听该组的讨论。此时，将一个特殊成员指定为主持人。主持人的任务是广播更新的盲密钥，以便整个小组在必要时更换其密钥。对于我们的示例来说，M_7 离开了这个小组，M_6 成为主持人。由于 M_7 知道保密密钥 K_0 和 K_2，因此它们必须被更换。这样，M_6 将选择一个新的随机数 na_6，它将用于更换 K_2。M_6 将向组内的每一个成员广播新的 BK_2，以便在其他成员仅仅更新组密钥 K_0 时 M_8 和 M_9 能够更新 K_0 和 K_2。图 9.8 给出了所需的改变。请注意，只有少量的密钥必须更换——这确实是一种十分有效的方法。

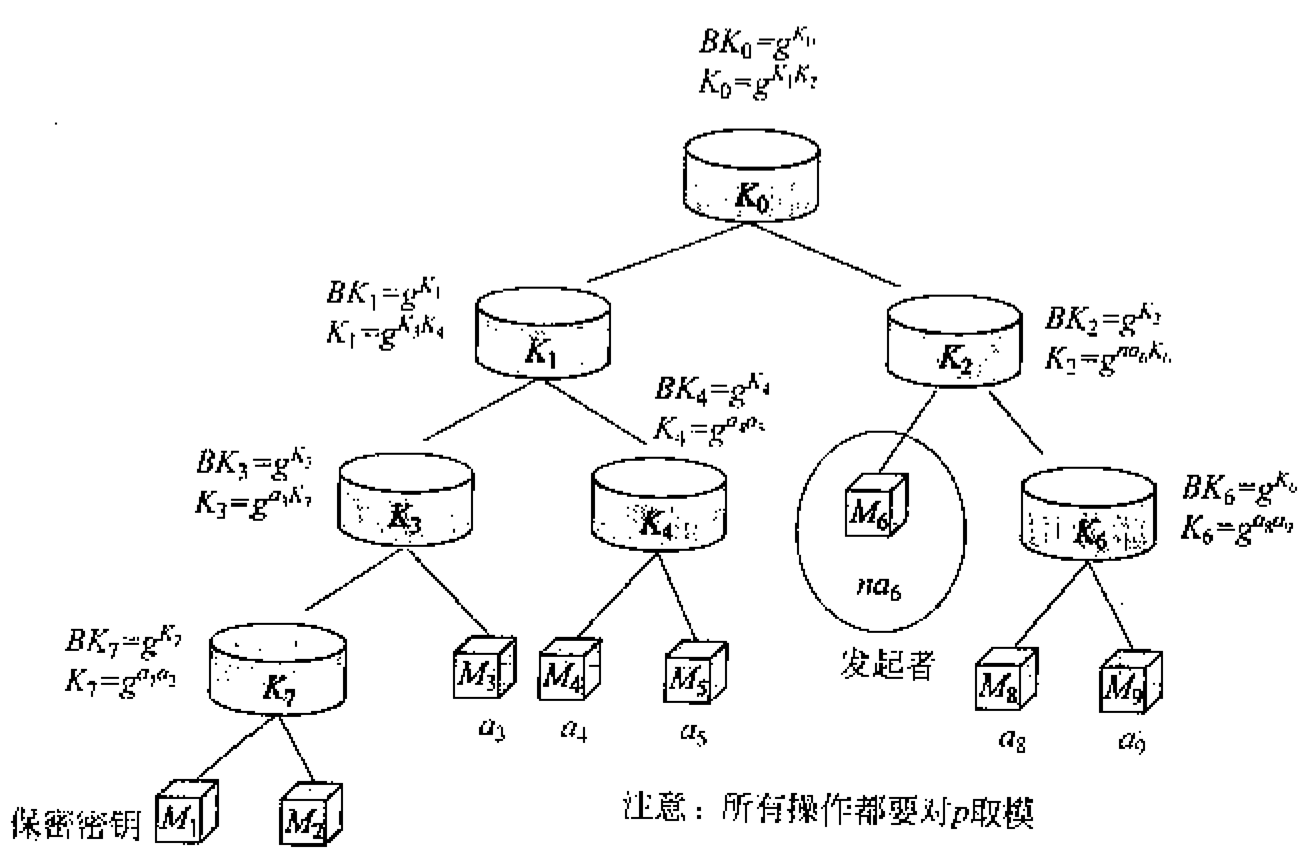


图 9.8　当一个成员离开时密钥的重新生成

向组中添加一个新成员也很容易。和前面一样，指定一个将更换盲密钥的主持人。例如，如果像图 9.9 那样 M_{10} 成为新成员，M_9 可以被指定为主持人。此时，M_9 将选择一个新的随机数 na_9，并使用它生成 K_8 和 BK_8，这些信息被传递到树结构中，以便创建新的组密钥。请注意，当 M_{10} 加入时，没有向他提供泄漏以前密钥的任何信息。从 M_{10} 到根节点路径上的所有保密密钥都被更换。

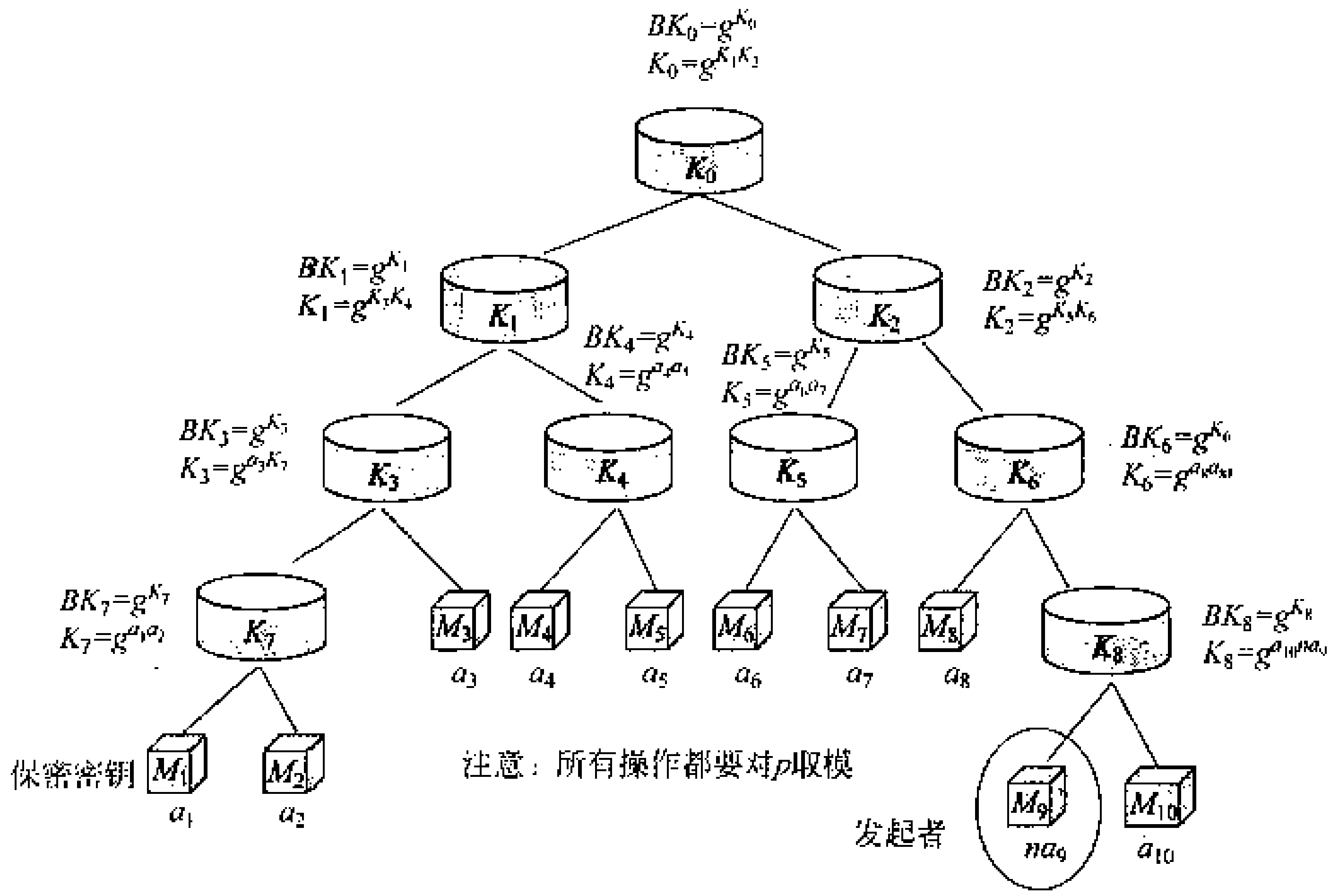


图 9.9 当一个成员加入时密钥的重新生成

9.1.3 广播加密

到现在为止，假定条件一直是多个参与方相互通信。他们来回地共享信息，以便建立密钥。但是，当不是这种情况时，该怎么办呢？当一方只能够发送，另一方只能够接收时，该如何建立密钥呢？这个问题看起来有点独特和编造的味道，但它却是一个相当普遍的问题。考虑一下付费电视的运营问题。家庭用户购买了一个解扰器，并把它连接到 TV 上。解扰器接收加密的 TV 信号，并进行解密以便观看。发送机构如何能够知道该使用哪一个密钥加密信号呢？家庭解扰器如何知道解密密钥呢？让这一切正常工作的过程称为广播加密。

由于通常只有部分用户被授权访问特定消息，因此广播加密问题被进一步复杂化了。在 TV 示例中，只有部分的解扰器用户可以请求特定的按次计费节目，因此只有这些用户能够解码 TV 信号。所有其他用户都不应该能够免费地观看这个节目。图 9.10 展示了想要观看"The Sun Sets at Noon"的观众的情况。

已经建议和实现了几种广播加密的方法。一个方法是为每一个 TV 盒分配一个它自己的私钥，之后，为每一个 TV 盒发送使用每个私钥加密的定制信号。对于数百万个家庭用户来说，这种方法要求相同的电视信号被加密和传输数百万次。一定应该存在更好的方法。

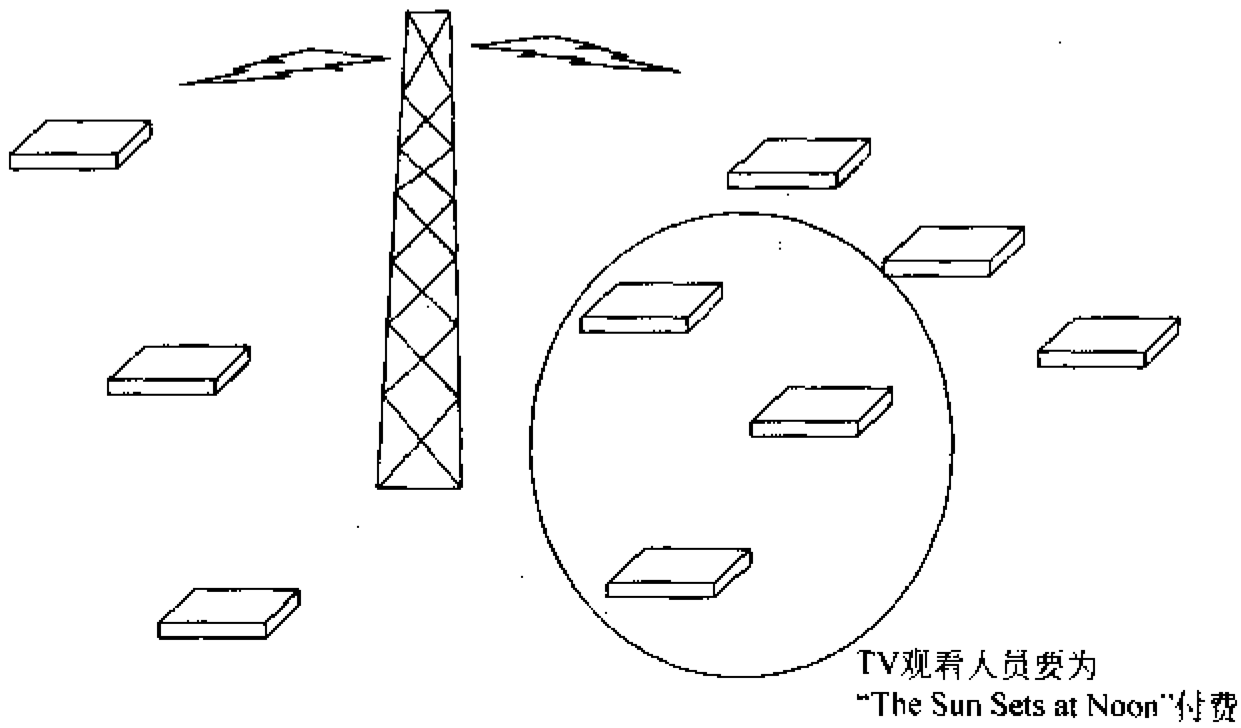


图 9.10　广播子集

正是这样，人们已经开发了一种类似于上一节介绍的 TGDH 协议的方法。给定 n 个设备，假设每一个设备都已经分配了一些保密信息（假设设备 u 的信息为 I_u）。设备是 n，$S_1,\cdots,S_m$ 各种子集的成员。每一个子集 S_j 都有一个长久有效的密钥 L_j。给定一组撤回设备 R，那么未撤回设备的集合为 $N\backslash R$，它由 w 个不相交的子集组成：

$$S_{i_1}, S_{i_2}, \cdots, S_{i_w} \quad \text{因此}, N\backslash R = \bigcup S_{i_j}$$

给定子集中的每一个设备都知道或者能够计算该子集的密钥 L_i（对第 i 个子集来说）。每一个消息都通过某个算法 F 使用会话密钥 K 进行加密。每一子集的会话密钥都通过某个算法 E 使用子集密钥进行加密。之后，消息以图 9.11 所示的形式发送给所有设备。头部标识子集并提供了加密形式的会话密钥，各个子集可以使用它进行解密。这样，只有那些授权接收消息的设备能够访问会话密钥。

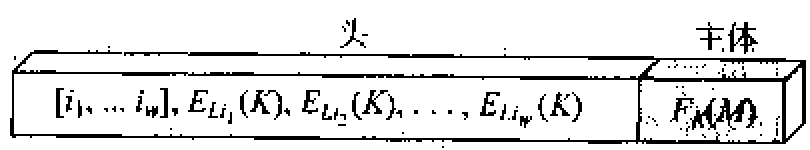


图 9.11　加密广播消息格式

在这个方法能够工作之前必须解决的问题是如何分配密钥，以便这些子集能够定义为只由授权设备组成。利用与 TGDH 协议很类似的树形表示能够完成这个任务。这种方法称为完全子树方法（complete subtree method）。构造一个完全二叉树，其叶节点标识设备，内部节点保存密钥值。图 9.12 给出了一个由 8 个设备组成的树的示例。每一个设备都有自己的密钥，并且都知道从根节点到设备节点路径上的所有密钥。例如，设备 3 知道密钥 L_9、L_4、L_1 和 L_0。在这个结构中，可以很容易地定义只包含授权设备的一个子集。例如，如果在播放像"Happiness Always Jumps the Tracks"这样的流行按次付费节目时，或许用户 1、2、3、5、6、7 和 8（除 4 之外的所有用户）愿意付出 50 美元观看这个电影。在这种情况下，授权设备可以定义为三个子组：$\{D_1, D_2\}$、$\{D_3\}$、$\{D_5, D_6, D_7, D_8\}$。如图 9.13 所示，这三个子集由三个密钥集合定义：L_3、L_9 和 L_2。

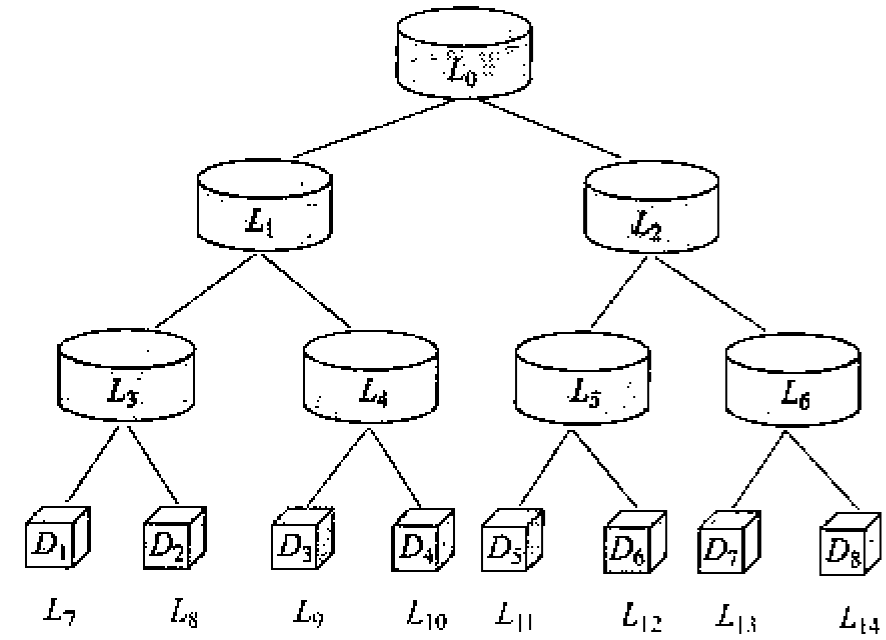


图 9.12　8 个设备的完整子树

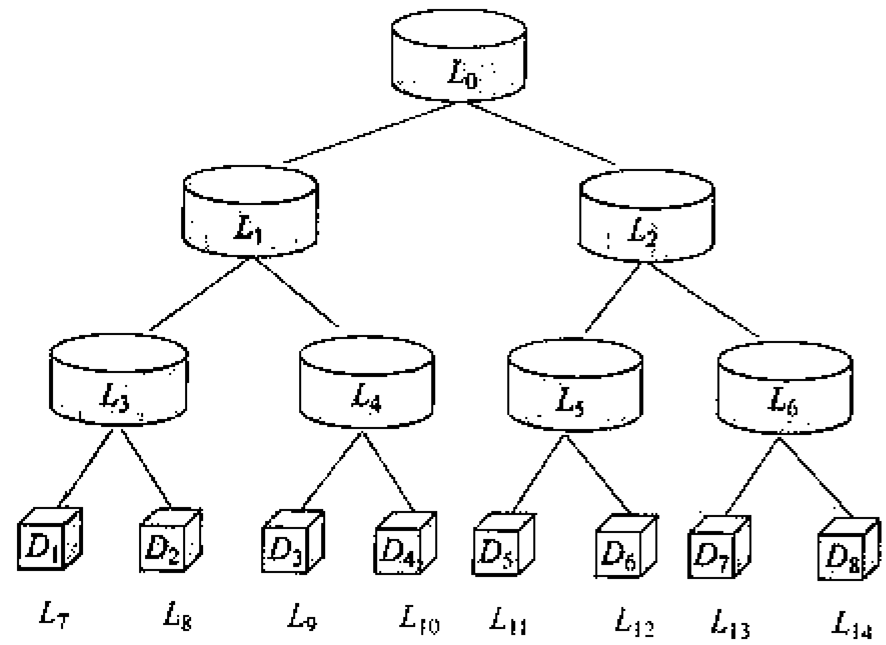


图 9.13　授权设备大组

即使有大量设备，由于每一个设备仅仅存储少量密钥，因此这种方法对于广播加密来说十分高效。例如，假设有 16384 个设备，这棵树也只有 16 层($\log_2 n$，这里 n 是设备的数量)，因此每一个设备只需要存储 17 个密钥。

在总结本节密钥管理之前，注意到我们几乎没有提到实际加密算法是很重要的。在介绍了八章如何加密和解密消息之后，应该把兴趣转移到如何处理和分发密钥。这里我们所要学习的内容是，在现代密码学中，如何处理密钥与选择什么样的加密法同样的重要。

9.2 可 靠 性

在圣诞节假日中，充满祝贺话语的电子邮件邮箱正像充满圣诞卡的普通邮箱一样。其差别在于祝贺语有时候包含了比表面上看起来更多的东西。伪造的、来自某个朋友的电子邮件有时候会告诉收件人登录看起来无害的站点，比如 www.funholidaycards.com，但这个站点实际上包含了恶意代码。

2002 年，即时通信(IM)用户被电子邮件欺骗，警告他们其系统已经被感染，要求他们

下载恶意代码。这些用户被指示到某个 Web 站点下载某个特别的软件，否则他们将会被 IM 系统屏蔽。当然，在用户实际下载这些伪造的软件之前，他们并没有被真正的感染。

还是在 2002 年，eBay 用户接收到了一封请求他们更新其金融信息的电子邮件。这些用户被引导到一个名称为 www.ebayupdates.com 的站点。这个 Web 站点拥有真正的 eBay 公司标记，并且看起来是合法的。但这只是伪装。

对于这三个事件来说，使用加密可能什么也做不了——毕竟没有什么东西被加密。但他们都作了一件相同的事情：这些邮件都由欺诈者发送。这就提出了数字文档和消息都存在的问题，即使它们已经被加密。你如何能够知道它们的可信性呢？实际上，这与 1587 年密谋反对伊丽莎白女王的那些人面临着相同的问题。收集到的部分反对苏格兰玛丽女王的证据是以她自己的加密法发送的虚假消息。

当 Bob 和 Alice 相互发送文档时，也面临这个问题。利用 Diffie-Hellman 或某个变体的密钥交换协议，Bob 和 Alice 可以相对地确认他们分享了保密密钥。因此，问题不再是保密，而是可靠性。可靠性问题有多种形式。最简单的一种形式是完整性——消息曾经被修改吗？还可能是这种情况，当 Alice 向 Bob 发送消息时，她并不关心 Eve 是否阅读了这个消息。她和 Bob 只想确保 Eve 没有修改消息的内容。此时，他们可以在消息的末尾添加一个消息认证码(MAC)。MAC 的值依赖于消息的内容和保密密钥。

例如，如果 Alice 想向 Bob 发送一段可信的明文消息，它或许会使用 DES 的密码块链接模式加密这条消息。由于她并不关心私密性，因此她并不向 Bob 发送密文，取而代之的是，她将密文的最后一块(她的 MAC)附加到明文消息的末尾。当 Bob 接收到这条消息时，他也运行 DES，并将最后一块与 Alice 发送的 MAC 进行比较。如果两者匹配的话，那么可以认为该消息完整无缺，否则，就可以认为 Eve 以某种方式修改了消息。由于 Eve 并不知道 DES 密钥，所以她不能生成有效的 MAC。

为了验证可靠性和机密性，Alice 会首先使用她自己的私钥加密明文，然后使用 Bob 的公钥解密结果。当 Bob 收到这条消息时，他首先使用自己的私钥进行解密，然后使用 Alice 的公钥恢复实际的明文。由于只有 Alice 知道自己的私钥，使用她的公钥能够恢复明文这个事实就是只有 Alice 能够生成这条消息的证据。由于只有 Bob 知道他自己的私钥，因此只有 Bob 能够解密出原始密文。这个过程如图 9.14 所示。

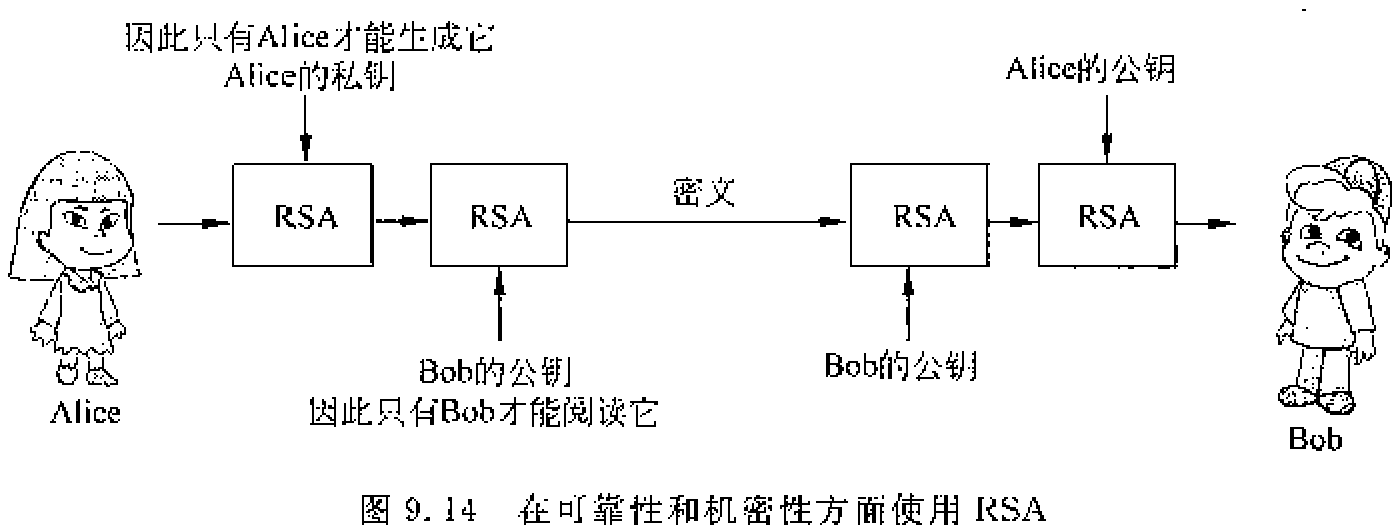


图 9.14　在可靠性和机密性方面使用 RSA

MAC 仅仅提供了一定程度的可靠性，也就是说，为了得到机密性，必须增加加密操作。法律方面经常要求比可靠性和机密性更多的东西：有的时候，Alice 和 Bob 之间发送的消息必须满足像正是签名文档那样的相同要求。这种需求导致了数字签名的开发。

9.3 数字签名

数字签名的目的和你自己的签名十分类似——它以法律可以接受的方式验证文档或消息的可靠性。事实上，美国国会和法院都已经同意数字签名的文档与手写签名的纸质文档具有相同的分量。

电子文档必须满足纸质文档具备的五个特性：无法伪造性、真实性、不可重用性、不可修改性和不可抵赖性。无法伪造性证明签名者，并且只有签名者能够实际签署文档。真实性说明接收人确信文档来自签名者。不可重用性要求签名只能属于一个文档，而不能应用于其他任何文档。不可修改性要求文档被签署之后就不能够被修改。最后，不可抵赖性确保签名者不能在以后声明他或她未签署过这个文档。

使用基于 RSA 的公钥协议(类似于上一节介绍的协议)能够满足这五个要求。但是，整个消息必须使用 RSA 进行加密，这种做法速度很慢。除此之外，由于存在对 RSA 某种形式的可能攻击(比如乘法攻击，后面将介绍)，因此人们更倾向于使用它来加密很短的文字。因此，数字签名通常是依赖于整个明文内容的一小段密文。给定消息 m，标准方法是创建一个定长的消息摘要 $h(m)$，之后签署摘要 $S[h(m)]$。所签署的消息以 $(m, S[h(m)])$ 对的形式发送。通过颠倒签名过程、恢复 $h(m)$ 的值，并将 h 应用到所接收到的消息上的方法验证消息。如果两个值相等，那么这个消息是真实的。这个使用过程如图 9.15 所示。

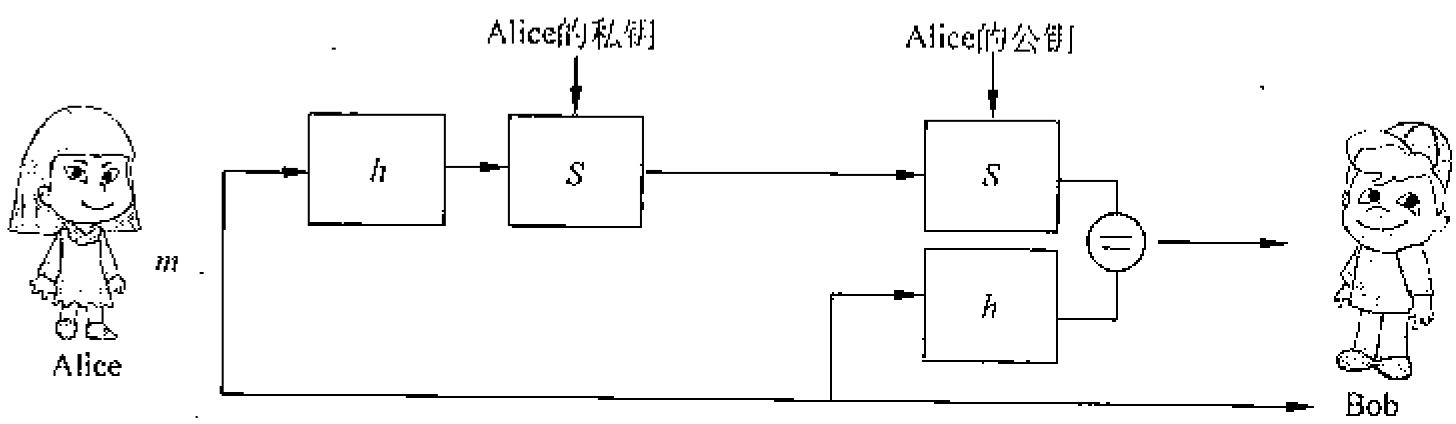


图 9.15 使用数字签名

你可能会感到奇怪，为什么要签名一个散列值而不是签名整个消息。换句话说，为什么 Alice 不使用她自己的私钥加密整个消息呢？不管怎么说，当 Bob 使用 Alice 的公钥解密消息时，不就证明这条消息来自 Alice 了吗？事实并非如此，原因在于这种类型的"原始"RSA 签名模式能够被乘法攻击所破解。它利用了特性：

$$(m_1 m_2)^d = m_1^d m_2^d \bmod n$$

如果 Alice 使用她的私钥签署了消息，并以形式 $s = m^d \bmod n$ 将它发送给 Bob，那么 Eve 能够对任何消息伪造她的签名。这是一个有力的陈述和严重的弱点。比如说 Eve 想在一个消息 f(来自单词 fake)上伪造 Alice 的签名。使用下述步骤它可以成功完成这个任务：

(1) 选择任意一个消息 m_1(一个在 $1 \sim n$ 之间的数)，并计算消息 $m_2 = f m_1^{-1} \bmod n$。这样，$f = m_1 m_2$。

(2) 欺骗 Alice 将 m_1 和 m_2 分别签名为 s_1 和 s_2。

(3) 然后对 f 的签名为 $s=s_1s_2 \bmod n$。

因此，Alice 必须首先散列她的消息，并对散列值签名。

在详细考察散列函数之前，值得注意的是，并没有任何东西要求只能使用 RSA 做数字签名。任何公钥算法都能够以类似的方式工作。例如，使用 ECC 公开和私钥，能够在图 9.15 中实现椭圆曲线数字签名。

9.3.1 散列函数

图 9.15 展示的模式中的函数 h 称为散列函数。散列函数将任何长度的输入转换为定长(更短一些)的输出。对于散列函数 h 有六个要求：

(1) 能够接受任何长度的消息作为输入。

(2) 能够生成较短的定长输出(~100 位)。

(3) 对任何输入都应该能够容易和快速地计算出 h。

(4) 应该是一个单向函数——难以或不可能反推。也就是说，给定 $h(m)$，恢复 m 应该十分困难。

(5) 应该能够抵抗弱冲突。当两个不同的输入生成相同的输出时，就产生了冲突。

(6) 应该能够抵抗强冲突——也就是说，找到两个有意义的消息 m_1 和 m_2，使得 $h(m_1)=h(m_2)$ 几乎是不可能的。

散列函数经常用于保护口令文件。你的用户口令被散列，并且只有散列值保存在计算机中。这种方法防止了某人读取口令文件并知道你的口令值。当你登录时，计算机散列你所输入的口令，并将该散列值(而不是实际口令)与存储的散列值进行比较。如果匹配，那么就授权你进行访问。

对于安全用途来说，从前面列表的第 5、第 6 个要求中可以推导出散列函数应该满足的三个一般特性为：

(1) 抗逆向特性——给定 $h(x)$，计算 x 十分困难。

(2) 抗第二逆向特性——给定 $h(x)$，很难找到一个 x'，使得 $h(x)=h(x')$。

(3) 冲突特性——很难找到两条消息 x 和 x'，使得 $h(x)=h(x')$。

1979 年，Merkle 基于数据压缩函数 f 建议了一个散列函数的通用模式。压缩函数接受两个输入——n 位长度的压缩值(CV)和 b 位长度的数据值(M)——并生成一个 n 位输出。Merkle 建议的内容是，数据值有消息块组成，对所有数据块进行迭代处理，其系统结构如图 9.16 所示。

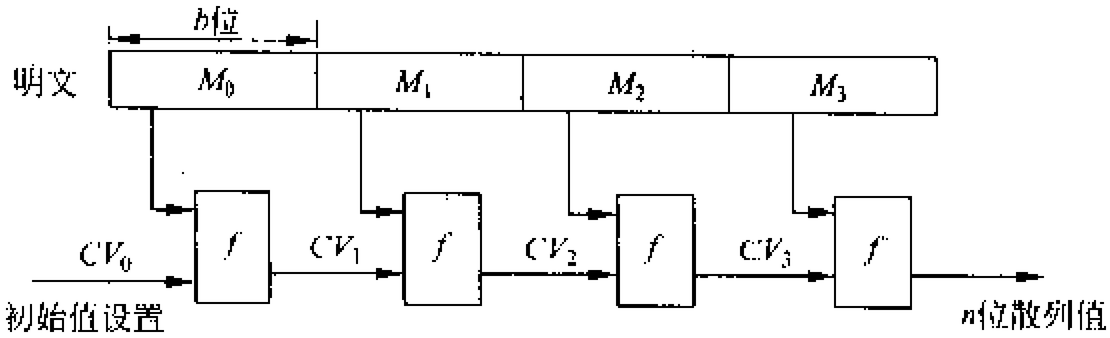


图 9.16　通用迭代散列模式

按下述过程，使用 XOR 运算可以构造一个简单的散列函数：将消息划分为固定长度的块，对每一块的第一个元素做 XOR 运算，得到第一个散列输出；对每一块的第二个元素做 XOR 运算，得到第二个散列输出；以此类推。例如，使用 8 位块，消息"Go now"(使用 ASCII 码)的散列值为：

G	01000111
o	01101111
n	01101110
o	01101111
w	01110111
	01011110 = 5E(十六进制)

只使用 8 位的块并不能生成抗冲突的散列值。在这种情况下，"bob"和"yes"具有相同的散列值(6F)。然而，更大的块大小可以提供更好的抗冲突能力，因此，Bob 和 Alice 决定采用这个散列函数。Bob 向 Alice 发送了一条消息，来回应收到了她的保密密钥："Alice，I just got your secret key."。他们很幸运，并不需要为他们的消息真正计算 XOR 值，原因在于 CAP 将为他们完成这一切。Bob 选中了 Signatures 菜单下的 XOR Hash 选项，将散列长度设置为 16 位，记录散列值(021F，如图 9.17 所示)，并和消息一起将它发送给 Alice。如果 Eve 截获了 Bob 的消息，并试图使用消息"Alice，I did not get your secret key."替换这个消息。当 Alice 计算这个伪造消息的散列值时(0D3E)，她就会检测到 Eve 的干扰。

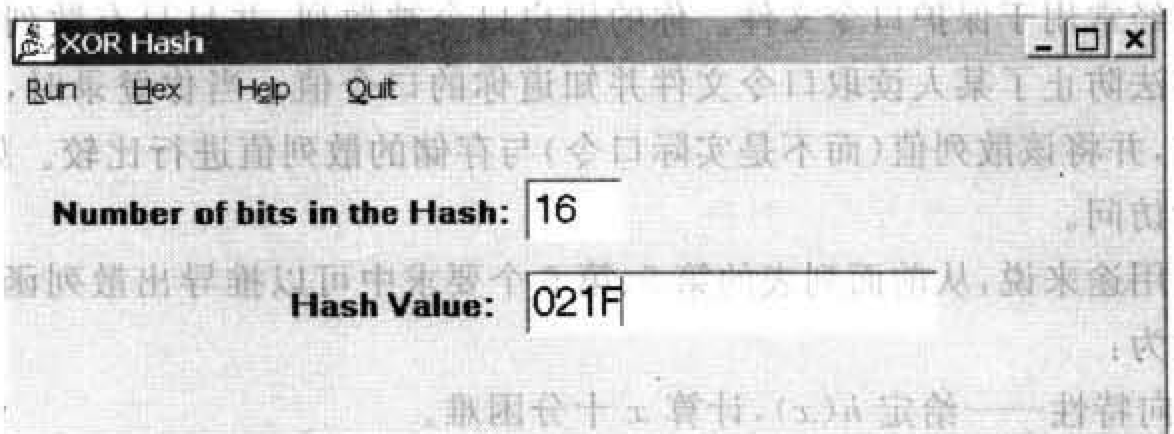


图 9.17　简单的 XOR 散列的 CAP 实现

9.3.1.1　MD5

简单的 XOR 散列函数并没有提供用作数字签名的足够的安全性。人们已经为此目的建议了几个更复杂的函数。其中之一是 Ron Rivest 开发的 MD5(消息摘要 5)算法。

MD5 接收任意长度的消息作为输入，并生成 128 位消息摘要作为输出。对于给定的长度为 L 位的消息，建立算法需要三个步骤。第一步是通过在消息末尾添加一些额外位来填充消息。填充是绝大多数散列函数的通用特性，正确地填充能够增加算法的安全性。对于 MD5 来说，对消息进行填充，使其位长度等于 448 mod 512(这是小于 512 位一个整数倍的 64 位)。即使原始消息达到了所要求的长度，也要添加填充。填充由一个 1 后跟足够个数的 0 组成，以便达到所要求的长度。例如，如果消息由 704 位组成，那么在其末尾要添加 256 位(1 后面跟 255 个 0)，以便将消息扩展到 960 位(960 mod 512＝448)。

第二步，将消息的原始长度缩减为 mod 64，然后以一个 64 位的数字添加到扩展后消息的尾部。在这个示例中，原始消息的长度为 704 位，其二进制值为 1011000000。将这个数

书写为 64 位数字(在开始位置添加 54 个 0),并把它添加到消息的末尾。其结果是一个具有 1024 位的消息。

第三步,MD5 的初始输出放在四个 32 位寄存器 A、B、C、D 中,这些寄存器随后将用于保存散列函数的中间结果和最终结果。初始值为(十六进制形式):

A = 67452301;　B = EFCDAB89;　C = 98BADCFE;　D = 10325476。

一旦完成了这些步骤,MD5 将以四轮方式处理每一个 512 位块。这个四轮过程如图 9.18 所示。每一轮都由 16 个阶段组成,每一轮都实现针对该轮的功能(F,G,H,I),对消息块部分作 32 位加法,对数组 T 中的内置值作 32 位加法,移位运算,最后作一次加法和交换运算。它真正地打乱了所有位。

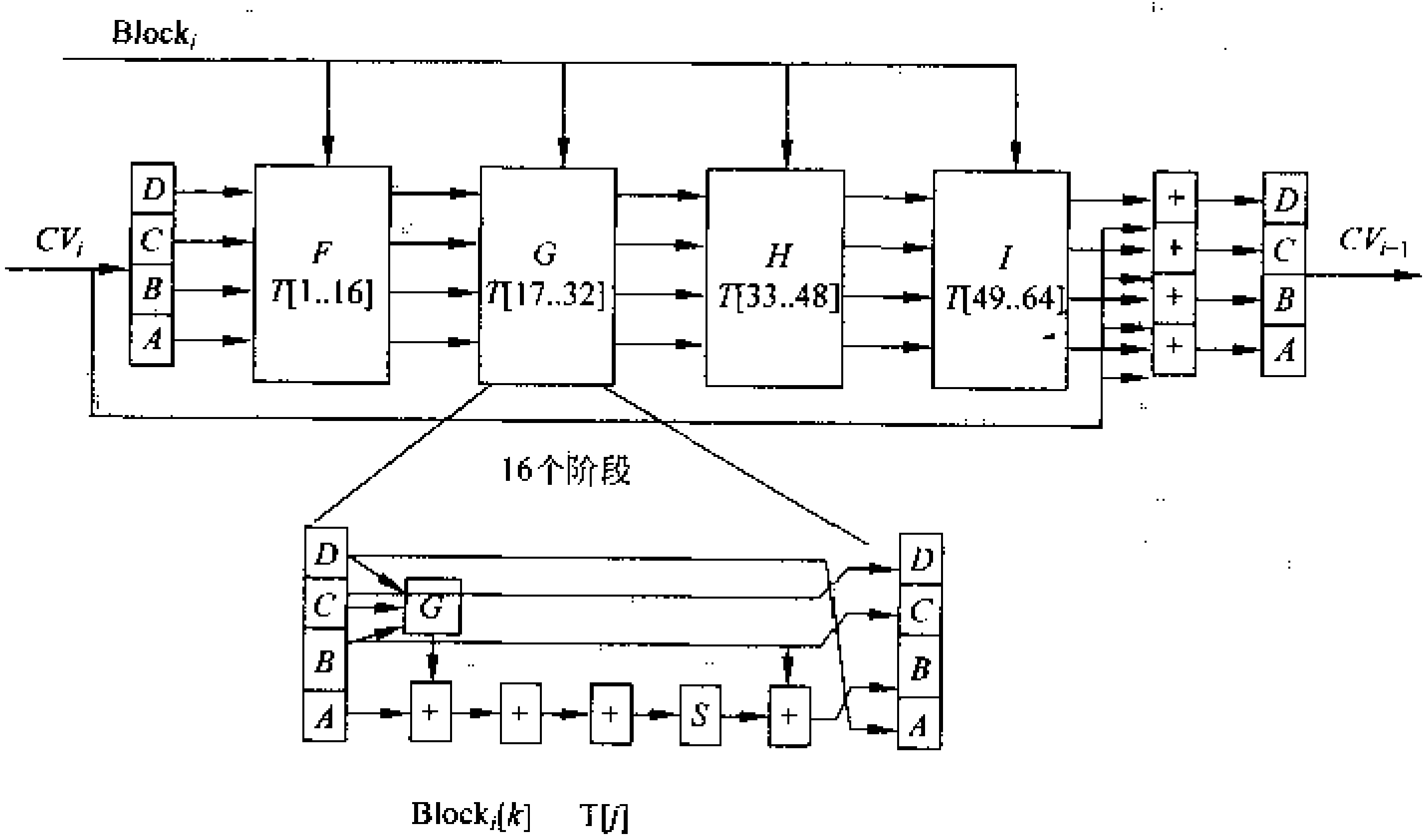


图 9.18　MD5 的各个轮及其阶段

特定轮功能接受三个 32 位字作为输入,并使用按位逻辑运算产生 32 位输出。图 9.19 所示的真值表列出了对应于每一个输入位组合的输出位值。32 位功能输出加到寄存器 A 的内容上(mod 2^{32})。

b	c	d	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

图 9.19　MD5 特定轮功能

512 位输入块被划分为 16 个 32 位字。在四轮中的每一轮内部,16 个字的每一个字精确地只使用一次;但是,它们的使用次序是不同的。对于第一轮来说(使用功能 F 的那一轮),输入块的 16 个字依次使用——也就是说,$block_i(j)$加到寄存器 A 上,这里 j 为当前的阶段(j 的范围为 1~16)。在第二轮,$block_i(k)$加到寄存器 A 上,这里 $k=(1+5j)$ mod 16,

其中 j 为当前的阶段。在第三轮，$block_i(k)$加到寄存器 A 上，这里 $k=(5+3j) \bmod 16$，其中 j 为当前的阶段。在第四轮，$block_i(k)$加到寄存器 A 上，这里 $k=7j \bmod 16$，其中 j 为当前的阶段。

之后，寄存器 A 中的新值模 2^{32} 与数组 T 中的常量元素相加。T 值由正弦函数导出，并罗列在图 9.20 中。之后，A 的结果值左循环移位，与 MD5 中的其他操作一样，循环移位的移位量在轮与轮之间和阶段与阶段之间都是变化的。每一轮的每个阶段的循环移位位数如图 9.21 所示。最后，将寄存器 B 加到寄存器 A 上，并进行寄存器置换。

T[1]	=	D76AA478	T[17]	=	F61E2562	T[33]	=	FFFA3942	T[49]	=	F4292244
T[2]	=	E8C7B756	T[18]	=	C040B340	T[34]	=	8771F681	T[50]	=	432AFF97
T[3]	=	242070DB	T[19]	=	265E5A51	T[35]	=	699D6122	T[51]	=	AB9423A7
T[4]	=	C1BDCEEE	T[20]	=	E9B6C7AA	T[36]	=	FDE5380C	T[52]	=	FC93A039
T[5]	=	F57C0FAF	T[21]	=	D62F105D	T[37]	=	A4BEEA44	T[53]	=	655B59C3
T[6]	=	4787C62A	T[22]	=	02441453	T[38]	=	4BDECFA9	T[54]	=	8F0CCC92
T[7]	=	A8304613	T[23]	=	D8A1E681	T[39]	=	F6BB4B60	T[55]	=	FFEFF47D
T[8]	=	FD469501	T[24]	=	E7D3FBC8	T[40]	=	BEBFBC70	T[56]	=	85845DD1
T[9]	=	698098D8	T[25]	=	21E1CDE6	T[41]	=	289B7EC6	T[57]	=	6FA87E4F
T[10]	=	8B44F7AF	T[26]	=	C33707D6	T[42]	=	EAA127FA	T[58]	=	FE2CE6E0
T[11]	=	FFFF5BB1	T[27]	=	F4D50D87	T[43]	=	D4EF3085	T[59]	=	A3014314
T[12]	=	895CD7BE	T[28]	=	455A14ED	T[44]	=	04881D05	T[60]	=	4E0811A1
T[13]	=	6B901122	T[29]	=	A9E3E905	T[45]	=	D9D4D039	T[61]	=	F7537E82
T[14]	=	FD987193	T[30]	=	FCEFA3F8	T[46]	=	E6DB99E5	T[62]	=	BD3AF235
T[15]	=	A679438E	T[31]	=	676F02D9	T[47]	=	1FA27CF8	T[63]	=	2AD7D2BB
T[16]	=	49B40821	T[32]	=	8D2A4C8A	T[48]	=	C4AC5665	T[64]	=	EB86D391

图 9.20 T 数组中 MD5 常量值

轮次	阶段															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	7	12	17	22	7	12	17	22	7	12	17	22	7	12	17	22
2	5	9	14	20	5	9	14	20	5	9	14	20	5	9	14	20
3	4	8	16	23	4	8	16	23	4	8	16	23	4	8	16	23
4	6	10	15	21	6	10	15	21	6	10	15	21	6	10	15	21

图 9.21 MD5 中位循环移位量

在完成所有四轮之后，ABCD 的初始值加到 ABCD 的新值上，生成第 i 个消息块的输出。这个输出用作开始处理第 $i+1$ 个消息块的输入。最后一个消息块处理完之后，ABCD 中保存的 128 位内容就是所处理消息的散列值。

9.3.1.2 安全散列算法(SHA)

NIST 于 1993 年开发的另一个散列算法称为 SHA。两年后，这个算法被修改为了今天广泛使用的形式。修改后的版本是 SHA-1，它是数字签名标准中要求使用的算法。

这个算法是在 MD4(MD5 的前驱)之后建立的模型，因此它也有一些自己的特点。它接受任何程度的输入消息，并生成 160 位的消息摘要(MD5 仅仅生成 128 位的摘要)。与 MD5 相似，这个算法操作划分为 32 位字的 512 位长度的块。它包含了四轮运算，每一轮有 20 个阶段。

第一阶段是填充消息，以便其长度达到 512 位的整数倍，填充方法与 MD5 的填充方法相同。SHA-1 的初始输入被放在五个 32 位的寄存器 A、B、C、D 和 E 中，这些寄存器随后将

用于保存散列函数的中间结果和最终结果。初始值(十六进制形式)为：

A = 67452301；　B = EFCDAB89；　C = 98BADCFE；　D = 10325476；　E = C3D2E1F0。

消息被划分成 512 位的块，每个块由 16 个 32 位字组成。利用下述参数，通过混合和移位，块中的 16 个字被扩充为 80 个字：

初始块——M_0，M_1，…，M_{15}

$W[0] = M_0$，$W[1] = M_1$，…，$W[15] = M_{15}$

$W[t] = M_{t-16}$ XOR M_{t-14} XOR M_{t-8} XOR M_{t-3}，　其中 t 的取值为 15 ~ 80。

与 MD5 使用 64 个常量不同，这个算法在各个阶段只加了四个常量值。这些常量值使用数组 K 定义，这里：

$K[t]$ = 5A827999　　t 为 0 ~ 19

$K[t]$ = 6ED9EBA1　　t 为 20 ~ 39

$K[t]$ = 8F1BBCDC　　t 为 40 ~ 59

$K[t]$ = CA62C1D6　　t 为 60 ~ 79

每个消息块都在四轮、20 个阶段中处理，它们显示在图 9.22 中。图中给出的两个循环移位被定义为：一个 5 位(S^5)的循环移位和一个 30 位(S^{30})的循环移位。每一轮都是用下表给出的不同函数：

b	c	d	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	0	1	0	1
0	1	1	1	0	1	0
1	0	0	0	1	0	1
1	0	1	0	0	1	0
1	1	0	1	0	1	0
1	1	1	1	1	1	1

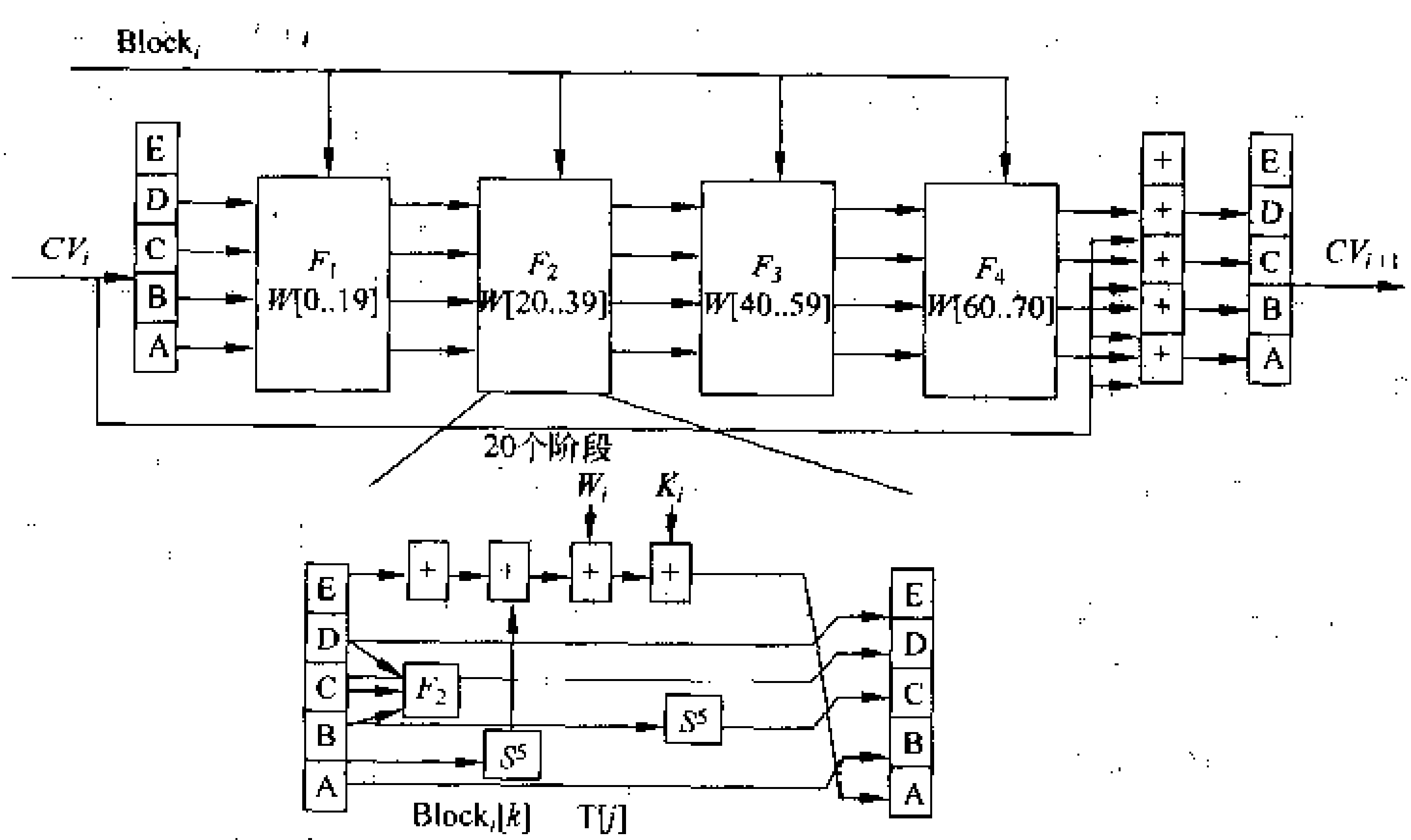


图 9.22　SHA 的各轮和各个阶段

SHA-1 既能够用于生成随机数，也能够用于散列消息。由于密码学中随机数有着重要的作用，因此值得考虑一下如何使用散列算法来构造随机数。

将 SHA-1 稍微修改一下，得到函数 $G(t,c)$。第一个参数 t 指定五个寄存器 A～E 的初始值；第二个参数 c 是要被 SHA-1 散列的消息。下面的算法将生成基于保密密钥 XKEY 的 m 个随机数和一组可选的种子值 XSEED_j（$j=0\sim m-1$）：

(1) 选择保密密钥值 XKEY，要生成的随机数的个数 m，每一个随机数的最大位数 b，（如果需要的话）一组 m 种子值 XSEED_j，以及一个 160 位的随机素数 q。

(2) 将 t 初始化为 67452301 EFCDAB89 98BADCFE 10325476 C3D2E1F0

(3) for $j=0$ to $m-1$, do

(a) $\text{XVAL}=(\text{XKEY}+\text{XSEED}_j) \bmod 2^b$

(b) $x_j=G(t,\text{XVAL}) \bmod q$

(c) $\text{XKEY}=(1+\text{XKEY}+x_j) \bmod 2^b$

x_j 就是随机数。

CAP 实现了这个算法：选择 Random 菜单项下的 SHA-1 Generator，打开如图 9.23 所示的窗口。生成的随机数可以直接使用，CAP 也能够把它们转换成能够在流加密法中使用的位流格式。

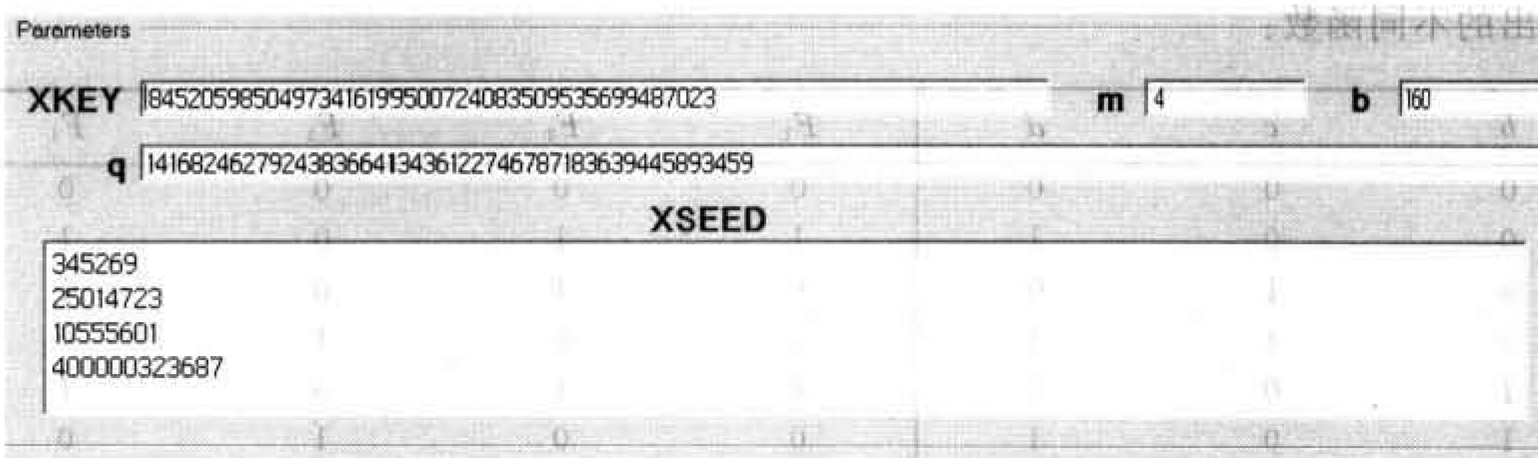


图 9.23　使用 CAP 生成随机数

在书写本文期间，发布了更新（并且更慢）版本的 SHA-2。SHA-2 设计用于满足像 AES 这样的新算法要求的更长位数。以三种版本提供 SHA-2，分别为 SHA-256、SHA-384、SHA-512。这些名称给出了位长度。例如，SHA-256 使用 8 个 32 位字，而 SHA-1 使用 5 个 32 位字（它只有 160 位）。在 SHA-384 和 SHA-512 中，字长被增加到 64 位。其他差别包括只有一轮、但阶段更多。例如，SHA-256 只有一轮，但由 64 个阶段组成。

9.3.1.3　MD5 与 SHA-1 的比较

由于 SHA-1 生成的消息摘要比 MD5 生成的消息摘要长 32 位，对于抗强力攻击来说它更安全一些。一般来说，人们认为，SHA-1 是抗密码分析的，而 MD5 可能是易受攻击的。上面两句间接说法隐含着，这两个算法的安全性是难以接受的。但不管怎么说，SHA-1 的步骤多些，但速度比 MD5 慢些。

9.3.1.4　基于块加密法的散列函数

SHA-1 和 MD5 都没有使用任何加密过程而从头建立。对于了解现代块加密法安全性和性能的人们来说，自然要问的问题是：“为什么不把块加密法用作散列函数的压缩操作

呢?”。例如，对于由 n 个块组成的消息来说，使用第 7 章介绍的任何一个加密法，都可以很容易地实现图 9.24 所示的模式。这个模式中，IV 是用于密钥的初始值，它用于加密第一个消息块。结果密文与消息块本身作异或运算，并将这个结果用作第二个消息块的密钥。这个过程持续进行，最后的输出就是散列值。实际上，有许多方法使用块加密法来构造散列函数。图 9.25 演示了其中的几种方法。实际上，1993 年，Preneel、Govaerts、Vandewalle 研究了 64 个这类基于块加密法的散列函数，发现只有 12 个函数是安全的。2002 年，Black、Rogaway 和 Shrimpton 修改了这 64 个模型，并发现其中有 20 个模型的安全性足以可接受。

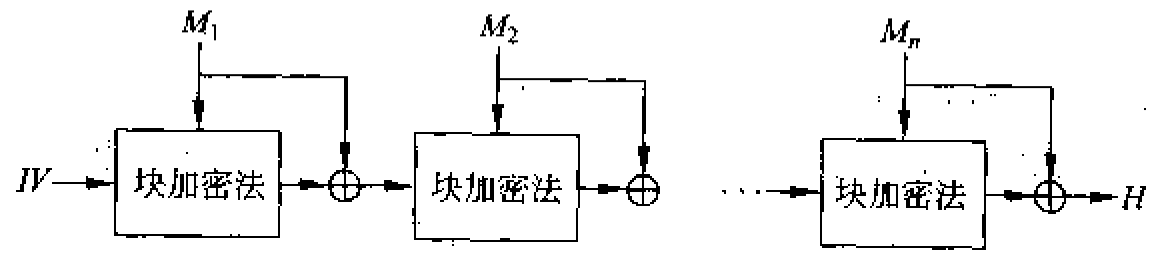


图 9.24　一个基于块加密法的散列函数

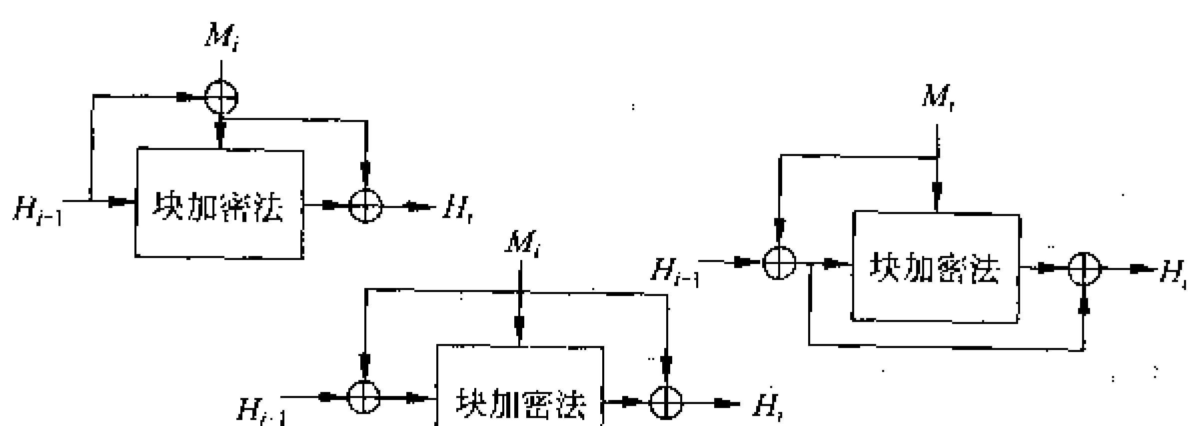


图 9.25　基于块加密法的其他散列函数

然而，存在各种各样的原因说明为什么不使用基于块加密法的散列函数。其中之一是像 DES 这样的块加密法只有 64 位的长度，它将散列函数的值限制为 64 位，对于安全散列值来说这样的值太小了。除此之外，加密算法的出口被美国政府严格限制，这意味着基于块加密法的散列函数同样受到这样的限制。最后一点是，对于有效地使用散列函数来说，典型的块加密法速度太慢。尽管存在这些问题，人们对使用 AES 依然存在某些兴趣，原因在于它具有更大的块长度，速度也相当快，因而克服了使用块加密法散列的主要问题。

9.3.1.5　对散列函数的攻击

由于散列函数用于确定消息是否被修改，因此，对散列函数攻击的目标是生成这样的修改后消息：其散列值与原始消息的散列值相等。例如，如果 Eve 找到了一对消息 M_1 和 M_2，使得 $H(M_1)=H(M_2)$，并且 Alice 愿意签署 M_1，但不愿意签署 M_2（即使 Eve 更喜欢 M_2），那么 Eve 就已经成功地实现了对散列函数 H 的攻击。如果 Alice 使用她的 RSA 私钥签署了 M_1，并向 Eve 发送了数值对 $(M_1, \text{Sign}[H(M_1)])$，Eve 就可能在法庭上声称她收到的是 $(M_2, \text{Sign}[H(M_1)])$，原因在于 M_2 的散列值与 M_1 的散列值相同。

Eve 的问题是如何找到具有相同散列值、并使 Alice 接受其中一条、反对另外一条的两条消息，她可以通过使用强力攻击作为开始。也就是说，她可以构造一组可以接受的消息 $A_1, A_2, \cdots, A_s$，和一组不可接受的消息 $R_1, R_2, \cdots, R_s$。之后散列所有消息，寻找具有相同散

列值的消息对。这看起来似乎是一个不可能完成的任务，但它说明，如果散列值比较小的话，Eve 将会取得成功。

这种类型强力攻击的可行性基于生日问题的解决（由于与人们的期待如此不同，因此有时候也称为生日悖论）。生日问题可以形式化地描述为：

在一个有 N 个人的房间中，N 应该必须多大才能够让两个人同一月的同一天过生日的概率大于 0.5？

绝大多数人会认为这个问题的答案是一个很大的值，但事实上这个值相当小。下面给出这个问题的解答：

(1) 对于进入房间的第一个人来说，有 365 个可能的不同生日。

(2) 当第二个人进入房间时，有 364 个与第一个人不是一天的可能生日，因此，不匹配的概率为 364/365。

(3) 当第三个人进入房间时，有 363 个与前两个人不是一天的可能生日，因此，现在不匹配的概率为：

$$\frac{(363)(364)}{(365)(365)}$$

匹配的概率为：

$$1-\frac{(363)(364)}{(365)(365)}$$

(4) 当 N 个人在房间中时，不匹配概率的一般公式为：

$$\frac{(365-N+1)\cdots(362)(363)(364)}{(365)^{N-1}}$$

匹配的概率为：

$$1-\frac{(365-N+1)\cdots(362)(363)(364)}{(365)^{N-1}}$$

(5) 使匹配概率大于 0.5 的 N 的值为 23。

生日问题一个良好近似解为 $N=1.17\times \mathrm{sqrt}(n)$，这里，$n$ 是可能的结果数（这种情况下，为可能的生日数，即 365）。生日问题可以用下面的、攻击散列函数的术语重新表述为：

给定一个生成 r 位消息摘要的散列函数，必须检查多少个消息，才能使得到具有相同散列值的两条消息的概率大于 0.5？

由于消息摘要为 r 位长度，总的消息数为 2^r。因此所需要的消息数量大约是 $2^{r/2}$。

如果 Alice 使用了一个生成 16 位散列值的散列函数，那么 Eve 所需要做的工作就是构造 Alice 可以接受的 $2^8=256$ 条消息和构造 Alice 不能接受的另外 256 条消息。存在 50∶50 的机会，使得这些消息中有两条消息生成相同的散列值。通过生成八个字不同的类似消息，就可以达到这个目的。例如，Eve 可以生成下述形式的可接受消息：

$$\text{I}\left\{\begin{matrix}\text{promise}\\\text{agree}\end{matrix}\right\}\text{to}\left\{\begin{matrix}\text{lend}\\\text{loan}\end{matrix}\right\}\left\{\begin{matrix}\text{25}\\\text{twenty-five}\end{matrix}\right\}\text{dollars to my}\left\{\begin{matrix}\text{good}\\\text{best}\end{matrix}\right\}\text{friend}$$

$$\text{Eve which she will}\left\{\begin{matrix}\text{repay}\\\text{return}\end{matrix}\right\}\text{to me in}\left\{\begin{matrix}\text{10}\\\text{ten}\end{matrix}\right\}\text{days or}\left\{\begin{matrix}\text{less}\\\text{sooner}\end{matrix}\right\}$$

$$\left\{\begin{matrix}\text{Yours,}\\ \text{Sincerely,}\end{matrix}\right\}\text{Alice}$$

她也可以生成下述形式的不可接受的消息：

$$\text{I}\left\{\begin{matrix}\text{promise}\\ \text{agree}\end{matrix}\right\}\text{ to }\left\{\begin{matrix}\text{give}\\ \text{offer}\end{matrix}\right\}\left\{\begin{matrix}\text{25}\\ \text{twenty-five}\end{matrix}\right\}\text{dollars to my}\left\{\begin{matrix}\text{good}\\ \text{best}\end{matrix}\right\}\text{ friend}$$

$$\text{Eve as a}\left\{\begin{matrix}\text{gift}\\ \text{present}\end{matrix}\right\}\text{ which she }\left\{\begin{matrix}\text{should not}\\ \text{will not}\end{matrix}\right\}\text{repay because I know}$$

$$\text{she needs this }\left\{\begin{matrix}\text{help}\\ \text{aid}\end{matrix}\right\}\left\{\begin{matrix}\text{Yours,}\\ \text{Sincerely,}\end{matrix}\right\}\text{Alice}$$

在散列了所有 512 条消息后，Eve 或许会发现，"I promise to lend 25 dollars to my best friend Eve which she will return to me in 10 days or less Yours, Alice"与"I agree to offer twenty-five dollars to my good friend Eve as a gift which she should not repay because I know she needs this aid Sincerely, Alice"有相同的散列值。如果 Alice 签署了可接受版本的消息，那么 Eve 就能够将它替换为不可接受版本的消息，原因在于这个消息生成了正确的散列值。

所有这一切都意味着，散列值的长度必须足够大，以便 Eve 不能够轻易地检查所有可能的消息。MD5 的散列长度为 128 位，这样，Eve 就不得不对每种消息类型都生成 2^{64} 条消息。SHA-1 的消息长度为 160 位，这样 Eve 就必须生成 2^{80} 条消息。两者都是不可行的。

9.3.1.6　CAP 中的散列函数

Bob 和 Alice 决定使用 MD5 代替 XOR 作散列，并且他们很幸运做出这种改变，原因在于 Eve 正在攻击他们。Bob 发送给 Alice 的前几条消息中的一条消息是"Alice, please meet with me next Tuesday in my office Bob"。Bob 以明文形式发送这条消息，并附加了 MD5 摘要值。由于 CAP 同时提供了 MD5 和 SHA-1，Bob 可以很容易地散列任何明文消息。Bob 在 CAP 的 Plaintext 框中输入给 Alice 的消息，选择 Signatures 菜单项下的 MD5，然后在 MD5 窗口中选择 Run(SHA-1 在 CAP 中以相同的方式实现)。其结果如图 9.26 所示。

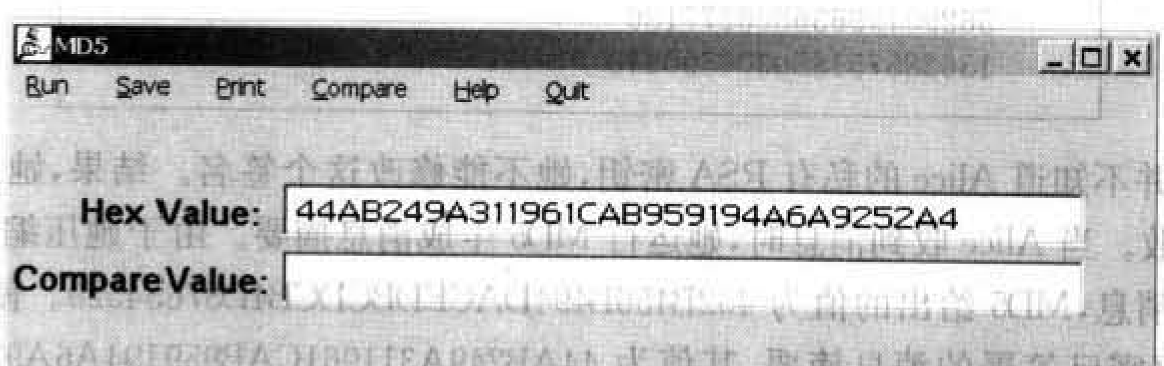


图 9.26　CAP 中的 MD5 散列

Bob 使用 CAP 中自己的 16 位私有 RSA 密钥签署这个消息：

$$N = 25953643153890769321l \quad d = 48960441661$$

之后，他发送携带下述已签署散列值的名文消息：

"Alice, please meet with me next Tuesday in my office Bob".
135594961498378250638
56082940190202137580
126036030298687927000
146359889510644548489
2438986244998163401
192894969220796395593
198148440850980822076
44547173078354731986
13764476474971611581
81178074635136373150
192894969220796395593
114438837427205627165
204086552991791154610
168969653468536651186
56290199636866277186
136285751850306590218

Eve 在消息到达 Alice 之前截获了这个消息。由于她不喜欢 Alice,她决定修改消息内容,从而让 Alice 失约。她向 Alice 发送了下述消息:

"Alice, please meet with me next *Wednesday* in my office Bob".
135594961498378250638
56082940190202137580
126036030298687927000
146359889510644548489
2438986244998163401
192894969220796395593
198148440850980822076
44547173078354731986
13764476474971611581
81178074635136373150
192894969220796395593
114438837427205627165
204086552991791154610
168969653468536651186
56290199636866277186
136285751850306590218

由于 Eve 并不知道 Alice 的私有 RSA 密钥,她不能修改这个签名。结果,她想让 Alice 陷入麻烦的企图失败。当 Alice 收到消息时,她运行 MD5 生成消息摘要。由于她压缩生成了 Eve 的、而不是 Bob 的消息,MD5 给出的值为 452B456B494DACEDBCDCB4D87654529。使用 Bob 的公开 RSA 密钥,她解密已签署的消息摘要,其值为 44AB249A311961CAB959194A6A9252A4。她使用 CAP 中可用的比较选项。当然,这两个值不匹配,提醒 Alice 与 Bob 联系。最终,她得到了正确的会面时间。

9.3.2 盲签名

有的时候,为了证明作者的身份,文档必须被第三方或证人签名,但文档内容必须保持私密性。在使用墨水签名的世界中,这个任务通常由公证人完成。在数字签名的世界中也

同样可以实现这个过程。

例如，Alice 想向她的银行发送一条经过公证的消息 m，但她并不希望其他人知道消息的内容。她找到当地公证人的 RSA 公钥——e 和 n。她选择 $1\sim n$ 之间的一个保密数字 b，并把她的消息转换为 $m_s=(b^e m) \bmod n$。Alice 携带其身份证明把 m_s 送给公证人。在验证了 Alice 的身份之后，公证人使用她自己的私钥签署 $m_{sign}=(m_s)^d \bmod n$ 并送走了 Alice。由于公证人并不知道 Alice 的保密随机数 b，他不能够恢复原始消息 m。由于银行也不知道 b，因此，在消息发送给银行之前，Alice 必须从 m_{sign} 中删除 b 的所有踪迹。Alice 必须在不破坏公证人签名的条件下完成这个工作。由于 Alice 不仅知道 b，而且也知道 $b^{-1} \bmod n$，因此可以很容易地完成这个任务。整个事件链条几乎像魔术一样：

$$m_{sign} = (m_s)^d \bmod n = (b^e m)^d \bmod n = b^{ed} m^d \bmod n = bm^d \bmod n$$

当 Alice 使用 b^{-1} 乘以 m_{sign} 时，其结果为发送给银行的消息的签名版本：

$$m_{bank} = b^{-1}\, m_{sign} = b^{-1}\, bm^d \bmod n = m^d$$

现在，银行可以使用公钥 e 验证公证人的签名，并使用 n 恢复原始消息 m 了。

CAP 提供了探索盲签名运算的工具。例如，Bob 想要发送消息“Alice，I am having this signed for verification Bob”。在明文窗口中输入消息后，选择 Signature 菜单项下的 Blind Signature 选项。他查找受信第三方的公钥，在 CAP 窗体中输入这个公钥，输入随机数，选择 Parameters 菜单下的 Find Inverse，最后选择 Blind Signature 菜单项下的 Create。结果如图 9.27 所示。当然，Bob 并没有受信第三方私钥的知识，即使它在 CAP 窗体中被输入。那里的 Private Key 选项是为了允许 CAP 完成盲签名过程中的所有角色。Bob 将盲签名消息发送给受信的第三方，并等待返回签名后的消息。

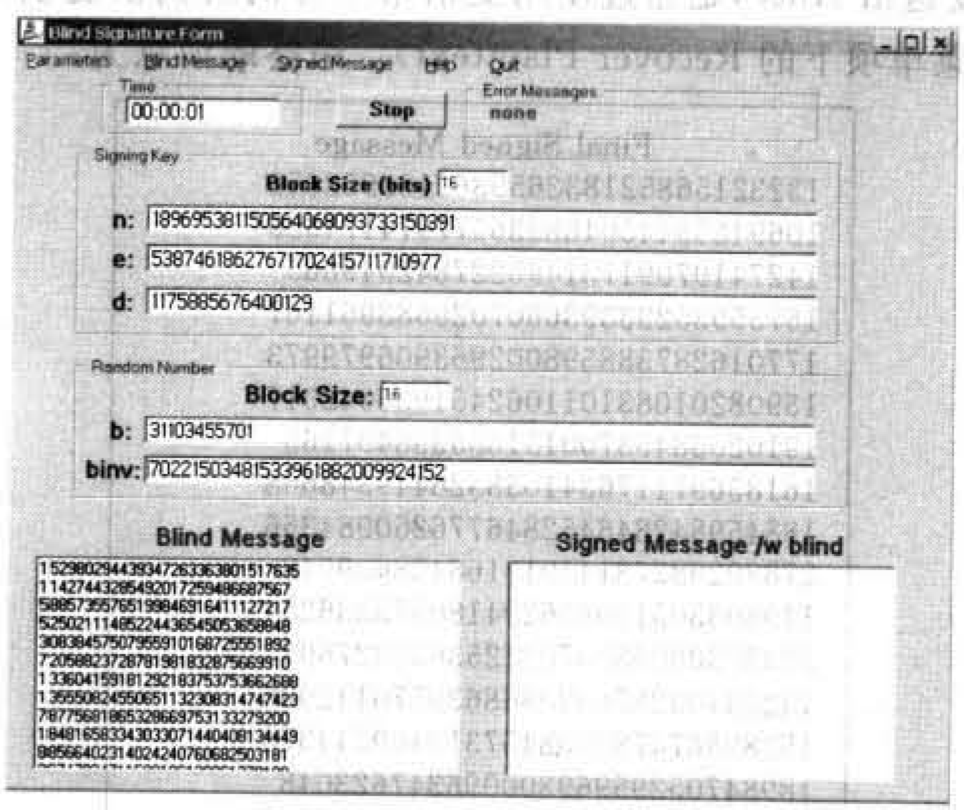


图 9.27　Blind Signature Form 窗口

受信第三方也会使用 CAP。盲签名消息被输入到窗体中，并选择 Signed Message 菜单项下的 Sign 选项。签署后的消息返回给 Bob，他将这个消息输入到 CAP 中，并选择 Signed Message 菜单项下的 Remove Blind 选项。下面是这个消息：

Signed Blind Message
5088848262898579755348489716O
7844352878234551661577050742l
72357145703599493969321919362
24542175058498502818925872318
82750544684084536246987294242
1128825050673333972315294615 85
11000435795679859037508082959 8
16609668399103376353733507809 0
11727338799789886311561970 08
29085308717752674522283189747
70338164817650700628985904758
1141942462454167709325117408 50
1246097632467637646075942871 83
1830161774996105074696323080 42
38786618645502052152908078528
96616995178790319762365231002
1851288395203906522106512491 52
1140610469632085516157919340 53
36583426924680986735380516868
4486326745865357073238838980
1677614956377899256706815394 97
33338739654911537992657221353
40588664019977608784068133481
1540641034697119536660372007 18
1033920955718956768703532666 13

最终的消息被发送给 Alice，她通过使用受信第三方的公钥恢复明文来验证消息。(选择 Signed Message 菜单项下的 Recover Plaintext)。最终的消息为：

Final Signed Message
1323215685218326593621099526 56
1069425944504889365779741743 28
14274197091731495227842049665
1675593523339366070296889611 67
1770162873885980029639069799 73
1590820108310110624519360420 17
19102058453700151886338407163
1618569711763410583264493486 68
1884598428484528467762509843 66
27889298278141013168428639948
1196033051808562941809733362 54
32330308003007021253630927697
1425340328760208686395701429 96
1528966747825584573134092443 70
18984705295969300096347623048
1428129165887260443588505276 90
1621654803152948746790933364 38
12449765676734525350863414512
1513172209039698641609528944 37
46479872139487594967761149225
70237624261547572956815196007
81527750113764252079703308517
47920912866286144714326193307
39596335900630012149462505421
29788317029354015869982415072

9.3.3 数字签名标准

正像有政府认可的标准加密算法(AES)一样，也有政府认可的数字签名标准，称为数字签名标准(DSS)。它于 1994 年被采纳，但依然笼罩在怀疑的阴云下，原因在于，与 AES 不同，其选择过程不是公开的。然而，DSS 是第一个被政府真正认可的数字签名系统，并且它也确实起到了取代 RSA 类签名的作用。

DSS 基于 El-Gamal 公钥系统；但是，它是一个严格的签名算法，并且不打算用于加密。正如在图 9.28 所看到的那样，它也是一个接近按不同观点签名理念的复杂(但依然很快)系统。例如，最后的验证测试基于值 r，它并不依赖于消息。其结果是，我们并不能够很明显地看出来要证过程到底是怎么完成的(像你将要看到的那样，它确实完成了验证)。

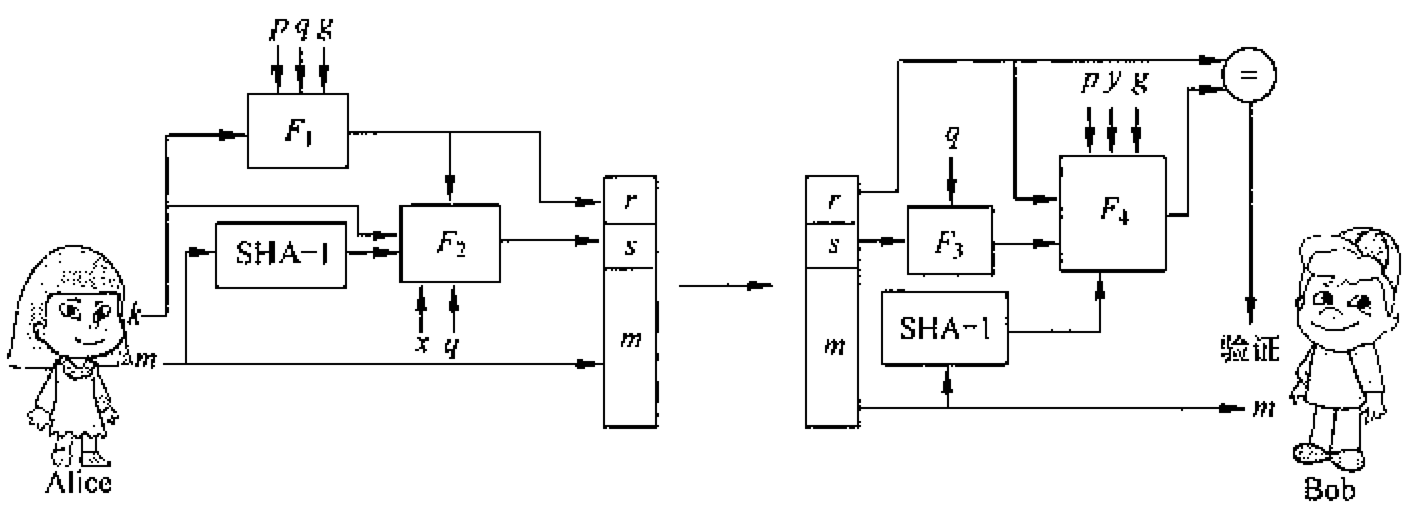


图 9.28　数字签名标准的运算

从图 9.28 可以看出，很清楚，DSS 使用了一大组参数：p、q、g、k、x 和 y。其中一些参数是公开的，另一些参数是不公开的，但它们都具有一些所要求的特性：

p 为公开的 1024 位素数；

q 为公开的、$p-1$ 的 160 位素数因子；

g 是公开的、1 mod p 的第 q 个根；

k 是私有的、160 位随机数；

x 是私有的、160 位密钥；

y 是公开的、512 位密钥，其中 $y=g^x \bmod p$。

DSS 中的四个函数使用这六个参数和消息的 SHA-1 散列值 h 来建立整个验证过程。附加在消息上的签名由值(r,s)组成，它们的定义为：

$$F_1\ r = (g^k \bmod p) \bmod q$$

$$F_2\ s = k^{-1}(h + xr) \bmod q$$

验证使用(r,s)的过程为：

$$F_3\ t = s^{-1} \bmod q$$

$$F_4 r' = (g^{ht} y^{rt} \bmod p) \bmod q$$

如果 $r=r'$，那么消息得到验证。

看起来几乎像魔法一样，但 DSS 能够工作，原因在于以下简单的数学推导：

$(g^{ht} y^{rt} \bmod p) \bmod q = (g^{ht}(g^x)^{rt} \bmod p) \bmod q$　　(使用替换 $y = g^x \bmod p$)

$$\begin{aligned}
&= (g^{(h+xr)t} \bmod p) \bmod q && \text{（指数合并）}\\
&= (g^{kst} \bmod p) \bmod q && \text{（因为 } s = k^{-1}(h+xr) \bmod q\text{）}\\
&= (g^{k} \bmod p) \bmod q && \text{（因为 } t = s^{-1} \bmod q\text{）}\\
&= r
\end{aligned}$$

与 MD5 和 SHA-1 一样，DSS 也带来了一些计算上的挑战。然而，CAP 提供了 DSS 的一个实现——选择 Signature 菜单项下的 Digital Signature Standard。打开如图 9.29 所示的窗口。菜单列表将引导你一步步地创建签名。例如，要创建公钥值 p、q 和 g，选择 Key 菜单下的 Create Global Keys 选项。在窗口中输入 q 所要求的大小，选择 Create Keys 选项，生成一组如图 9.30 所示的全局密钥。利用全局密钥、私钥、消息的散列值，主 DSS 窗口将生成两个散列值 (r,s)。它还提供了一个测试选项，以便验证消息散列的有效性。

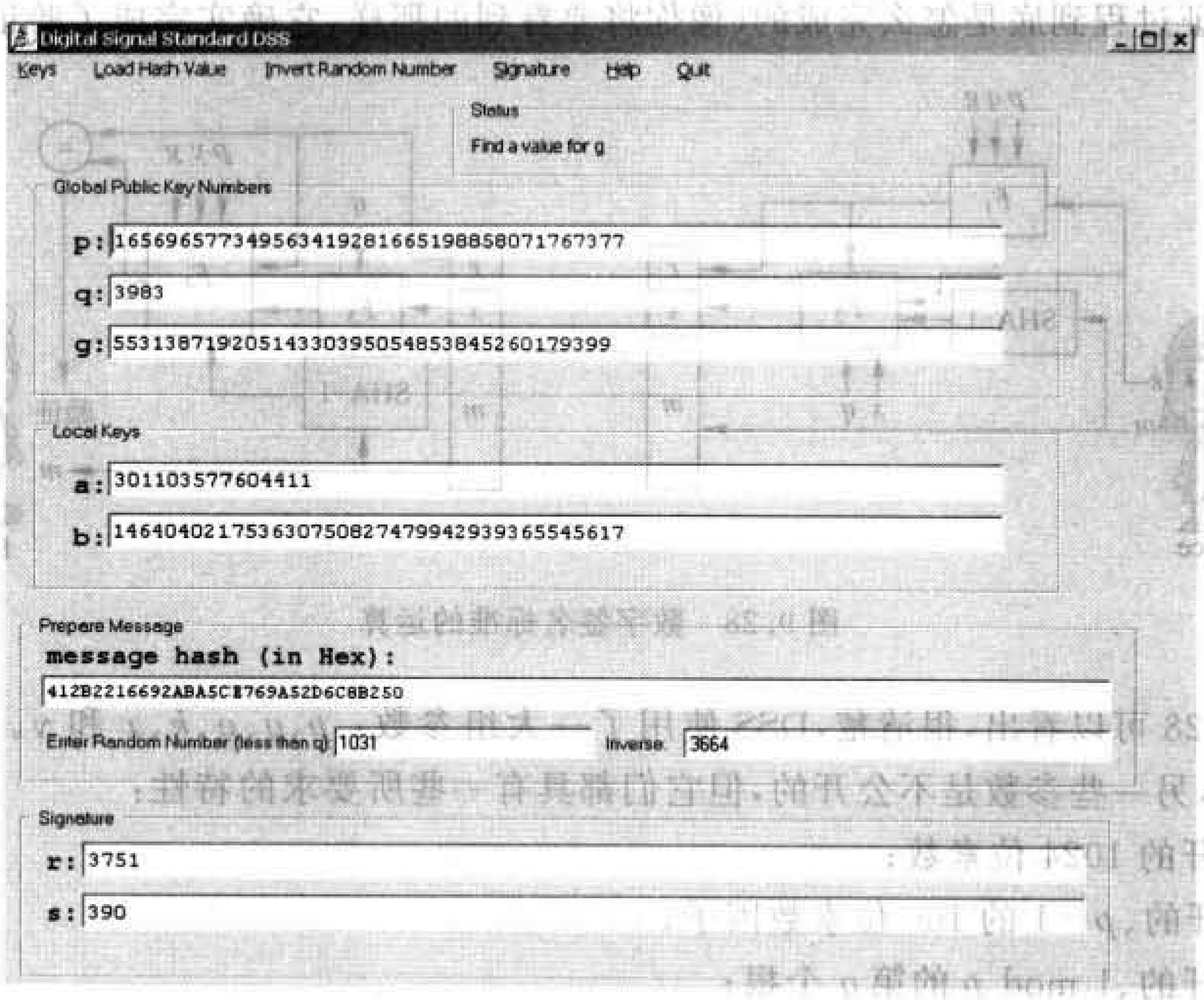


图 9.29　CAP 中的 DSS

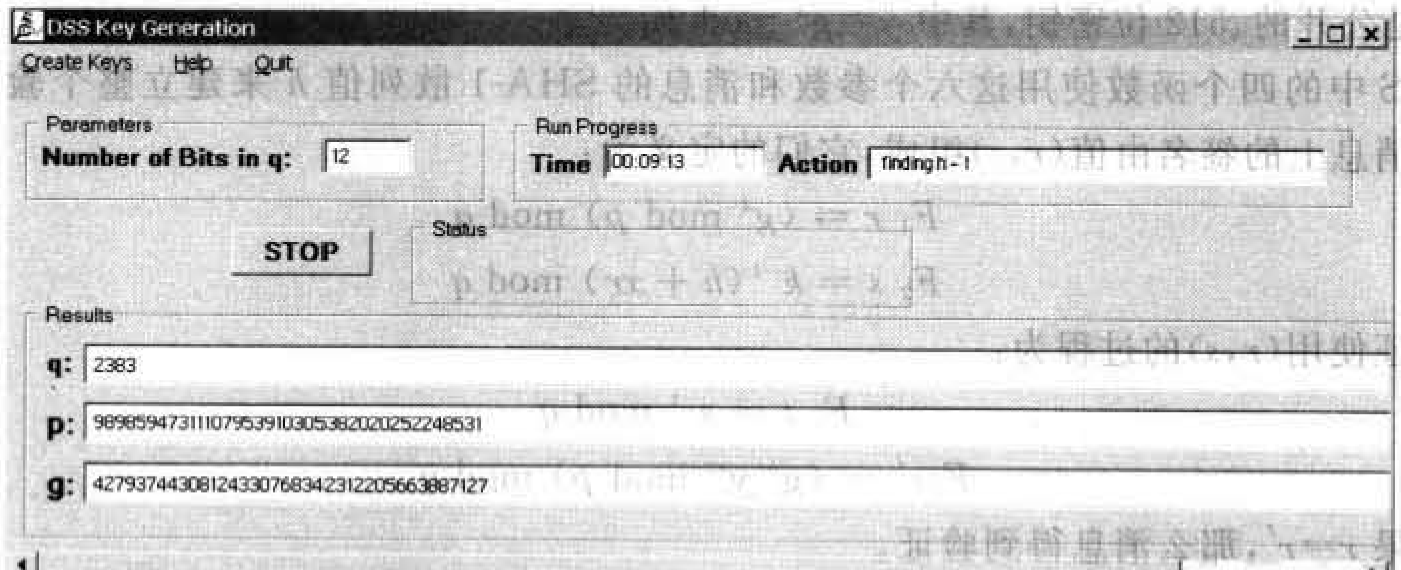


图 9.30　CAP 中的 DSS 密钥生成

1994 年认可的原始 DSS 于 2000 年被修改了一下，这样，现在就有了三种官方认可的数字签名方法：DSA、RSA 和 ECDSA。正如所预计的那样，RSA 数字签名方法使用 RSA，但只应用于散列后的消息（请谨记，原始的 RSA 签名受到了乘法攻击）。除此之外，散列后的消息被填充，以便改进抗 RSA 已知攻击的能力。使用 RSA 公钥(n,e)对消息 M 作 RSA 签名的形式化过程为：

(1) 使用 SHA-1[$h=\text{SHA}(M)$]计算 M 的散列值。

(2) 创建填充字符串 $M=6|PS|h|33CC$，这里，PS 是十六进制值“BB…BA”的字符串，这样选择 PS 的长度，使得填充后整个字符串的长度与 n 的位数相同。

(3) 使用私钥 d 计算 $s=M^d \bmod n$。

(4) 发送带签名 s 的原始消息 M。

在这个模式中，最后四个填充十六进制值（33CC）标识散列函数为 SHA-1。如果使用不同的散列函数，那么就在末尾添加不同的四个十六进制数字代码编号。

椭圆曲线数字签名算法（ECDSA）（现在已经是 DSS 的一部分了）包括类似的步骤。先是 ECC 公钥和私钥。已知值为曲线的次数(n)、基点(P)、私钥(d)、公钥$(Q=dP)$，以及消息 M。SHA-1 用于创建散列后的消息[$h=\text{SHA}(M)$]。创建消息摘要的后续步骤为：

(1) 选择随机数 k。

(2) 找出基点 $kP=(x,y)$，并计算 $r=x \bmod n$。

(3) 计算签名 $s=k^{-1}(h+dr) \bmod n$。

(4) 和消息 M 一起发送签名(r,s)。

使用下述步骤验证这个签名：

(1) 计算 $w=s^{-1} \bmod n$。

(2) 找到 $u_1=hw \bmod n$ 和 $u_2=rw \bmod n$。

(3) 找到点$(x,y)=u_1P+u_2Q$。

(4) 计算 $v=x \bmod n$，如果 $v=r$，那么签名被验证。

9.4 公钥基础设施和证书

数字签名和公钥系统通常都会受到与 Diffie-Hellman 密钥交换系统十分相似的中间人攻击方法的攻击。例如，Bob 想向 Alice 发送一条消息，但并没有记住它的密钥。由于 Alice 已经将她的公钥发布到 Internet 上，以便任何人都可以向她发送安全消息，因此 Bob 登录保存那个公钥的站点并复制这个公钥。为了确保消息不被修改，他是用自己的私钥加密消息，之后他想到的事情是将 Alice 的公钥用于最后一层的保护。然而，Bob 和 Alice 都不知道的是，Eve 已经使用她自己的公钥修改了那个发布的密钥。当 Bob 向 Alice 发送加密的消息时，Eve 截获了消息。她先用 Bob 的公钥、再用自己的私钥恢复消息并阅读消息。使用 Alice 的真正密钥（Eve 在更换发布的密钥时就保存了这个密钥），Eve 重新构造消息，并把它发送给 Alice。由于 Alice 的私钥将能够解密 Eve 截获的消息，并在最后阶段 Bob 的公钥也能够正常工作，Alice 相信这条信息是安全的。这个过程如图 9.31 所示。

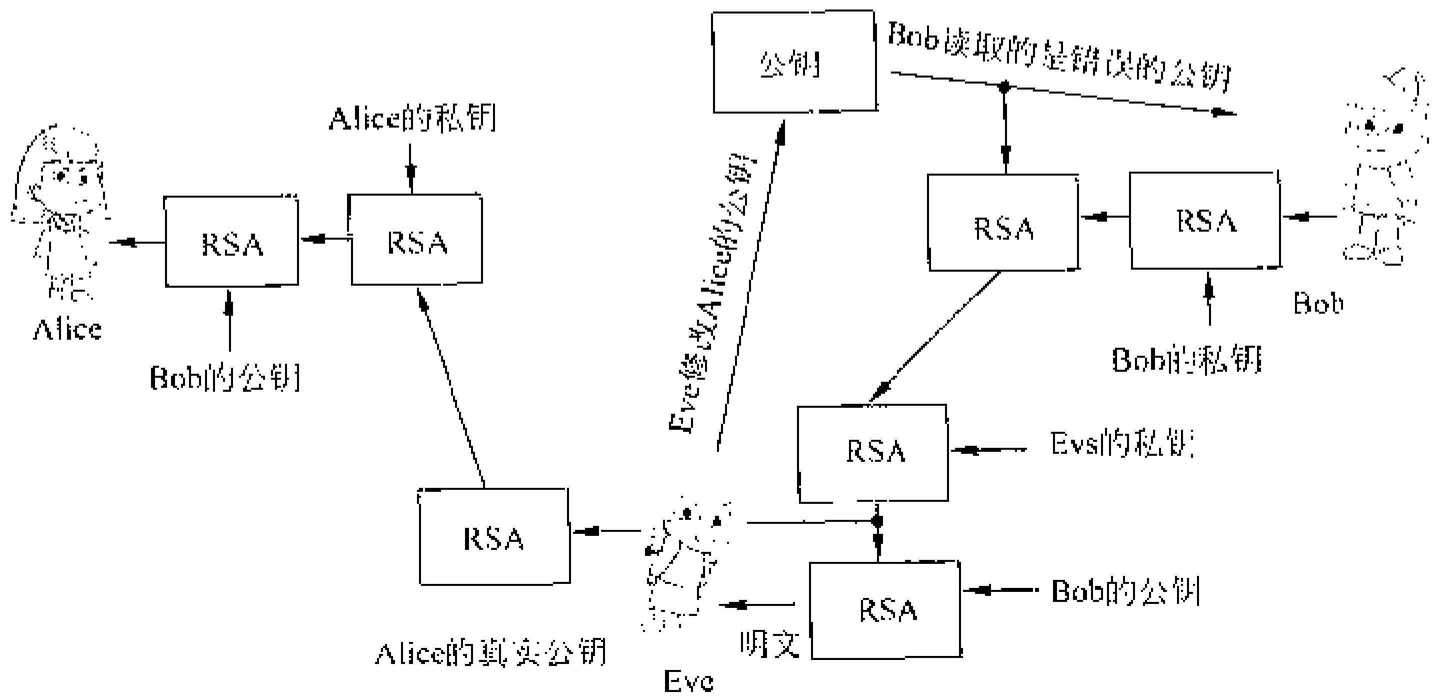


图 9.31　对数字签名协议的攻击

Bob 和 Alice 所遇到的问题应该是信任问题。他们可以相互信任，但他们如何能够知道与其通信的人到底是不是对方所声称的人呢？Bob 和 Alice 如何才能够知道他们所受到的公钥真正属于他们要发送消息的人呢？如果只是 Bob 和 Alice 之间的通信，那么它们可以使用诸如 Diffie-Hellman 之类的密钥交换过程。但事实上，他们属于大团体的一部分，他们希望自己的公钥能够被信任并在团体中使用。他们需要一种建立信任的机制。

公钥基础设施（PKI）的角色就是在公钥系统的用户之间建立起高度信任。它通过提供发布公钥的安全方法来达到这个目的。PKI 的两种基本操作是：

证明（将公钥值与所有者绑定的过程）；和

验证（验证证书依然有效的过程）。

PKI 由几部分组成：数字证书认证中心（CA）；数字证书注册审批机构（RA）；存储库和归档库。数字证书认证中心是运营 PKI、得到信任的第三方。CA 发行证书、跟踪老的或无效的证书、并维护一个状态信息归档库。RA 为 CA 验证证书的内容。存储库是用户可用的证书数据库。

9.4.1　建立证书

Bob 和 Alice 都决定他们需要有一个证书来维护对自己公钥的控制。Bob 首先采取行动。他的努力过程如图 9.32 所示。他选择了一个 CA，并发送了一个请求证书的申请。CA 向他返回一张申请表。Bob 填写了申请表，并按照说明，使用作为申请一部分、由 CA 提供的软件生成一对密钥。客户端软件生成 Bob 的新密钥、将私钥保存在 Bob 机器中的某个安全区域，并创建一个口令以便 Bob 能够访问密钥。之后，Bob 签署并提交填完的申请表，它包含了给数字证书注册审批机构的新公钥。数字证书注册审批机构验证 Bob 的身份（后面详细介绍），并创建一个数字证书请求，然后发送给 CA。当 CA 生成证书、使用其私钥签署证书并把证书发送给 Bob 后，整个过程完成。

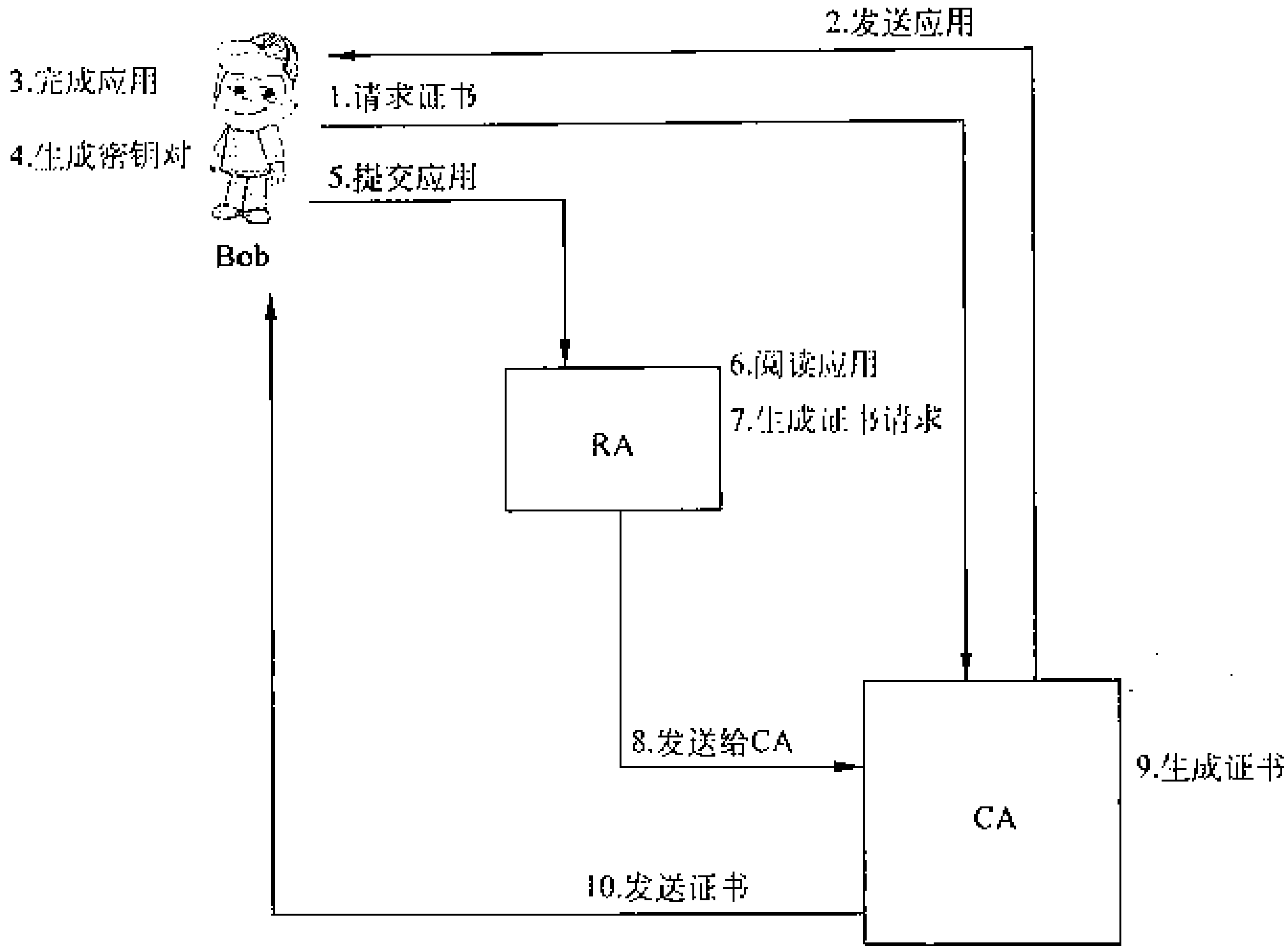


图 9.32 PKI 操作

理解这个过程的细节是重要的，原因在于证书的有效性取决于用户在 CA 中的信任层次。例如，Bob 和 Alice 都必须信任 RA 验证其身份的方法。不同的 RA 有不同的方法。有些 RA 要求申请人携带自己的身份证明亲自到本地办公室并领取其公钥。另外一些 RA 则要求申请人向他们传真申请人的驾驶证或其他证明文件以及申请人的公钥。这种情况下，RA 将首先向申请人发送一个使用申请人公钥加密的申请 ID。申请人必须解密该 ID，并在其他文档中包含这个申请 ID。

9.4.2 证书内容

由于存在几家 CA，因此，如果它们都生成类似的证书才是有用的；否则的话，用户就会在寻找证书的必要信息过程中感到困惑。为此，人们已经建立了 X.509 国际标准，以便指定有效证书的内容。

这个标准由 10 个域组成，其中的一些域是可选域。第一个域是证书格式版本号——当前有三个版本的 X.509(版本 3 是最新版本)。其后是证书序列号，它是一个由 CA 分配的惟一编号，以便确保不会发行重复的证书。下一个域是签名算法域，用于指示 CA 使用的签名算法和公钥加密算法。第四个域是证书发行机构的名称。这个名称由另一个国际标准给出，它指定了国家编码和 CA 组织代码。第五个域是有效期，它指定了证书首次成为有效证书的日期和证书过期日期。用户名称(Subject X.500 Name)包含在第六个域。用户最感兴趣的域是第七个域，它包含了用户的公钥并标识了用户使用的算法。下两个域是可选域，它由证书发行的惟一标识和用户惟一标识组成。这些域用于两者中存在重名 X.500 名称的情况。最后一个域是 CA 的签名。图 9.33 展示了一个证书示例。

Version	2
Serial Number	10035
Signature Algorithm	SH-1,RSA
Issuer DN	cu=US,o=VF
Validity Period	04/01/2003 08:00:00 04/01/2005 05:00:00
Subject DN	c=US,o=gov cn=John Smith
Subject Public Key	RSA,5503 3997...
Issuer UID	Usually omitted
Subject UID	Usually omitted
Signature	6A21 3E9F...

图 9.33　X.509 证书格式

9.4.3 使用证书

一旦 Bob 和 Alice 都收到了他们的证书，他们就做好了相互通信和与 Internet 世界的其他部分通信的准备。图 9.34 演示了 Alice 和 Bob 之间一次典型通信的示例。Alice 向 Bob 发送了一条使用她的私钥签署的消息。当 Bob 收到消息后，他从 CA 那里请求 Alice 的公钥。CA 将使用 CA 私钥签署的 Alice 的证书发送给 Bob。Bob 使用 CA 的公钥验证证书上的签名。这样，Bob 确信他收到了 Alice 的公钥，而不是收到了其他第三方（比如 Eve）的密钥。Bob 使用 Alice 的公钥打开她的签名，并验证该消息确实发自 Alice，并且在传输过程中没有被篡改。

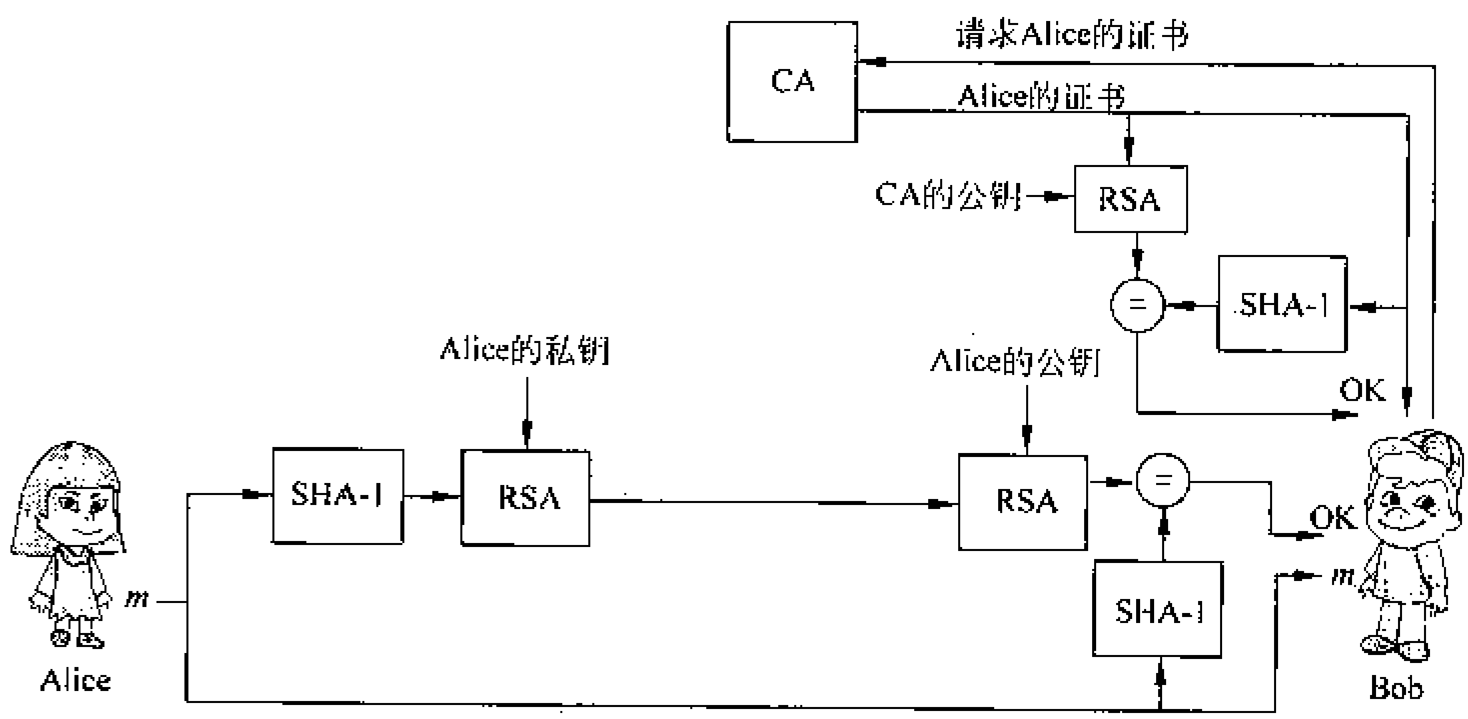


图 9.34　使用数字证书

由于 Alice 的公钥正确地解密了她的签名并且她的签名与 Bob 收到消息的散列值匹配，所以他相信消息的内容。他信任消息来源的原因是 CA 验证了 Bob 使用的公钥确实是 Alice 的公钥。看起来这个过程比他们之间相互通信的过程要复杂。但在当今的世界上，我们需要能够信任我们所收到的消息，这有时也意味着事情要略复杂一些。

9.4.4　证书的回收

由于某个可疑或已经觉察到的危害，有时候证书必须在他过期之前被收回。可能的原因包括证书所有人辞去了她的工作并且不再属于某个公司及相应的 CA，或许发现了明显的证书滥用。可能访问在有效期内这个证书的任何人都需要知道它已经被收回。有几种方法可以完成这个任务。一般来说，CA 维护着一个证书回收列表(CRL)，它包含了所有回收的、CA 签署证书的一个带时间标记的列表。CRL 可能每小时更新一次、每天更新一次或每周更新一次。周期性地检查这个列表是证书用户的责任。

9.5　应　　用

在前面的章节中已经介绍了现代加密法的几种应用，比如在无线等效隐私(WEP)协议中 RC4 的使用、在 IPSec 协议中 Diffie-Hellman 的使用。很显然，现代加密法的应用范围比这里提到的要广泛得多。这些算法使用的细节或许最好留给一本网络安全教材去讲解；但是，还是值得考虑一下其他应用，特别是在包含完整加密法协议的那些应用。

9.5.1　智能卡

几乎每一个人都有(或者都看见过)信用卡。它是一个一面写着名字和编号，另一面上有一个磁条、钱夹大小的塑料卡。在欧洲，绝大多数磁条卡已经被智能卡取代；在美国，替换过程才刚刚开始。正像图 9.35 所示，智能卡上有一个嵌入在塑料片中的微型电子设备，它上面有六到八只引脚。这个设备实际上是一台拥有自己 CPU 和内存的计算机。

典型的智能卡设备有一个 8 位的处理器、串行输入输出、用于保存程序数据的 ROM、存储计算数据的 RAM 和一个操作系统——所有这些东西都放在面积小于 $25mm^2$ 的区域上。某些智能卡还拥有类似 FPGA 的设备，称为 EEPROM，它能够存储定制的数据-程序。智能卡微处理器芯片的布局如图 9.36 所示。

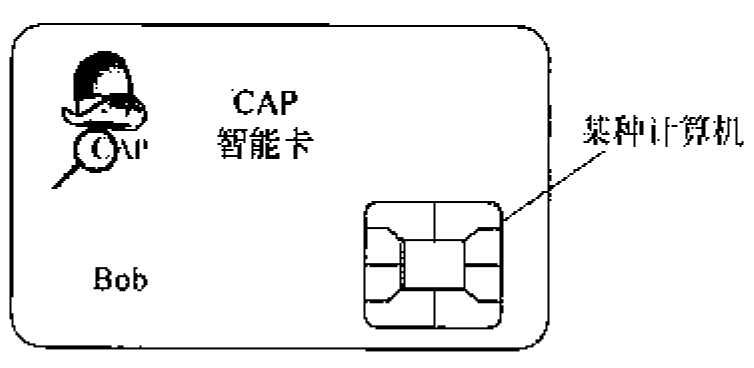


图 9.35　典型的智能卡

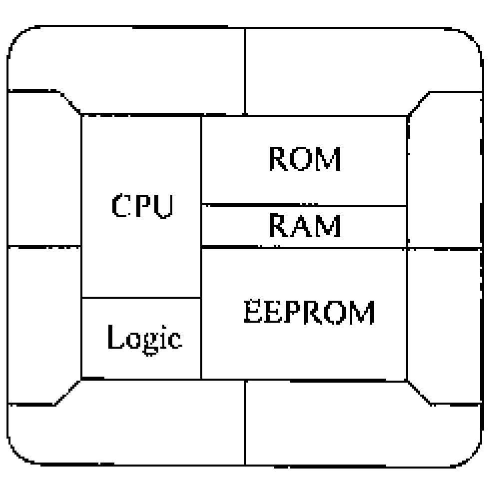


图 9.36　典型智能卡微处理器芯片

当智能卡在日本首先引入时，它从 1970 年开始了自己的发展。今天，世界各地每年制造的智能卡超过十亿张，并且其数量还在快速增长。

有三种类型的智能卡。像图 9.35 所示的接触型卡必须插入导读卡器中。非接触卡通

过使用感应线圈收发无线电频率信号而增强了能力。通过在读卡设备附近滑过(通常在两三英寸内)这种类型的卡而读取其内容。第三种类型是组合卡,它既可以插入读卡器中,也能够通过无线电射频(RF)信号读取。

智能卡有三项主要功能:(1) 信息存储和管理;(2) 持卡人标识;(3) 运算(特别是加密-解密)。这些功能被转换为各种各样的应用,比如以电子钱包的形式存储货币;存储口令和加密法密钥;或者加密和解密数据。这些东西都与密码学有着明显的关系。

例如,智能卡可以用于网络登录。代替输入用户名和口令登录到网络的传统方法,智能卡的登录过程可能要求首先将卡插入导读卡器;然后显示一个如图 9.37 所示的登录窗口。智能卡以加密格式存储复杂口令,但用户所有记住的东西是一个简单的 PIN 值。其最终结果是,为了访问网络,要求用户实际拥有智能卡和用户名与 PIN 的知识。

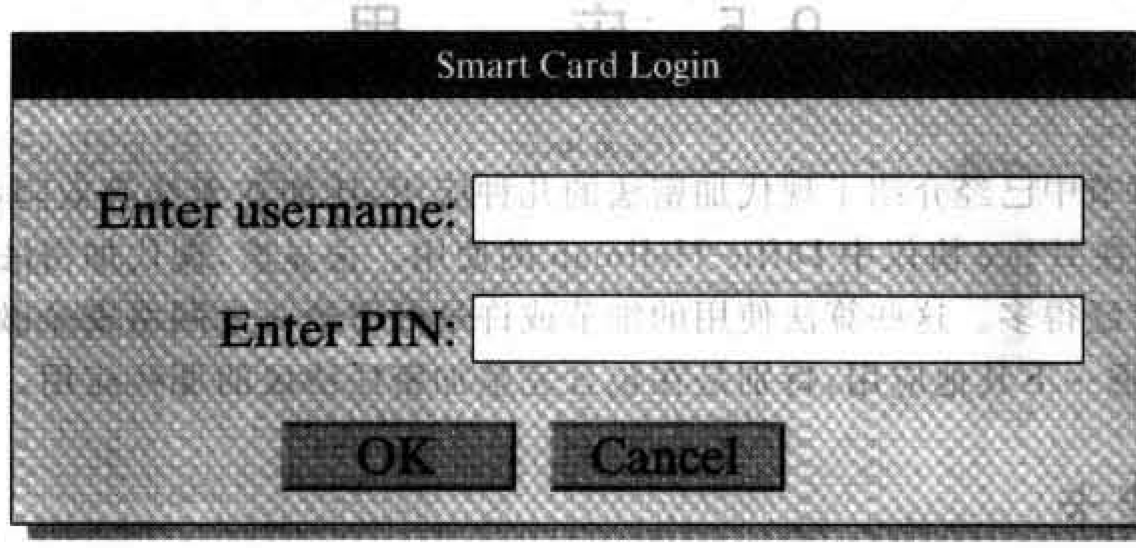


图 9.37 智能卡登录窗口

智能卡不仅仅只能存储口令。对于给定的处理能力,他们能够被用于签署诸如电子邮件这样的文档。实际上,智能卡是安全存储数字签名所需私钥的优秀设备。例如,图 9.38 演示了使用智能卡签署文档的过程。该文档被散列,其散列值发送给读卡器。智能卡使用私钥和散列值签署文档,并将签名返回给系统。

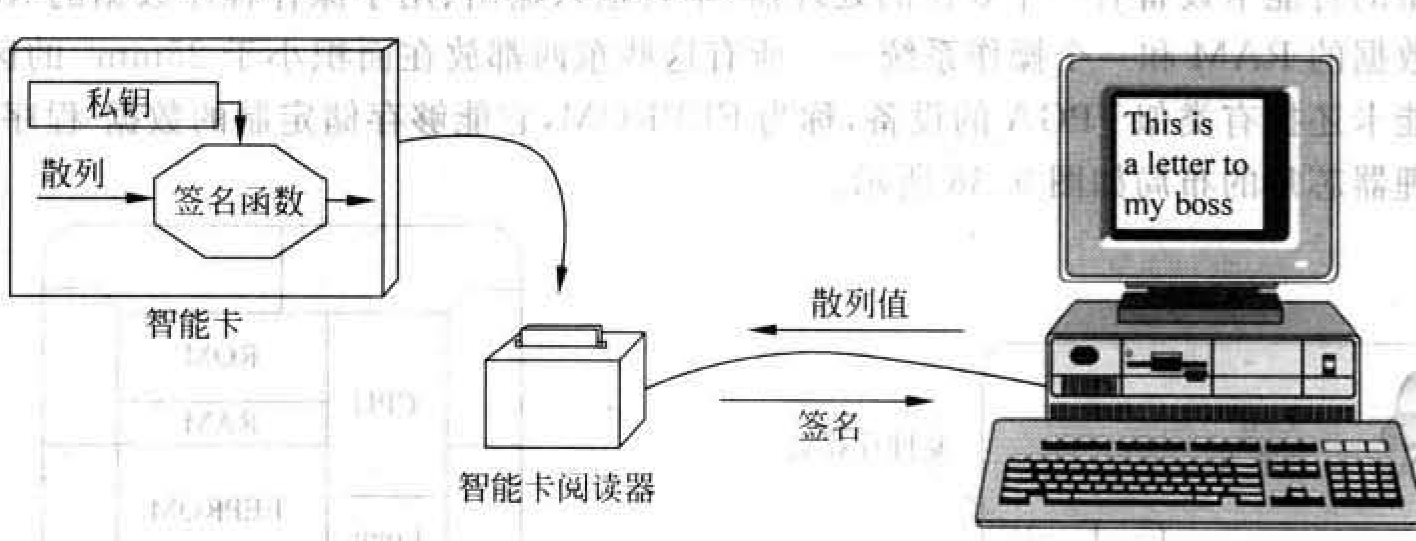


图 9.38 基于智能卡的数字签名

9.5.2 安全套接层

回顾一下第 7 章(准确地说是 7.6 节),那里讨论了 IPSec 中块加密法的使用,但许多细节被留到以后讲述。好了,现在就是以后了。IPSec 设计用于在网络层提供安全连接,在这里,网络包使用块加密法加密。本节中,通过考察安全套接层或称为 SSL 来讲述安全 Web

连接的细节。SSL 的任务是在传输层保护应用的安全(参考 7.6 节,复习一下分层结构)。事实上,一个经过些许修改版本的 SSL 称为 TLS,即传输层安全。

SSL 最初由 Netscape 公司开发,用于提供 Web 事务的认证、完整性及私密性。它借助于两个协议完成这些任务:SSL 记录协议和 SSL 握手协议。

SSL 握手协议启动这个过程。它建立一条由一组通用算法和密钥保护的安全连接。图 9.39 展示了 Bob 如何与一个 Web 站点建立 SSL 事务的过程。Bob 向服务器发送一条"Client Hello"消息。这条消息包含了 Bob 所支持的加密法部件(密钥交换方法、块加密法、MAC 算法……)的信息。服务器以选择了本次会话所支持加密法部件的"Server Hello"消息作为回应。如果服务器拥有一份公开证书,那么它将与其公钥一起将证书分送给 Bob。Bob(实际上是它的浏览器)将验证证书,并检查确定 CA 的签名是否有效。如果一切检查通过的话,Bob 将生成一个一次性的、惟一的会话密钥,并使用服务器的公钥加密这个密钥。现在,Bob 确信了两个事实:(1) 服务器的身份已经被验证;(2) 双方都有会话密钥的一个副本。



图 9.39 一次简单的 SSL 握手

实际上,SSL 协议比 Bob 登录 Web 服务器的简单示例更为灵活。"Client Hello"消息提供了几种可能的加密法套件之一。可用的加密算法通常包括 DES、三重 DES、RC2、RC4 和 IDEA。

SSL 响应算法选择的方法带来了一个这里需要仔细说一下的重要问题,也就是向美国境外出口加密算法的问题。它起因于将加密算法归类为军需品,并因此受到严格出口限制。如果你曾经对自己购买的某些软件为什么警告说它只适用于美国国内感到奇怪的话,现在你知道为什么了。似乎你正在阅读的、本文中的材料也被美国政府认为与建立一个核武器计划是等价的。对于我们社会中信息的重要性来说,一个好案例就可以说明这种等价性。它不仅仅说明以合乎道德和伦理方式使用你已经学习的东西是如何的重要。从 SSL 的观点看,这意味着它有两个版本:一个能够在美国使用的强版本和一个可以出口到海外的弱版本。最初,弱版本限制为 40 位加密,现在扩展为 56 位加密。强版本通常使用 128 位加密。

"Client Hello"消息还提供了使用 RSA 还是 DSA 作为数字签名算法与使用 MD5 还是 SHA 作为散列函数的选择。而且,SSL 可用的密钥交换方法也有多种。Bob 使用 RSA 作为前述的密钥交换方法,但 SSL 也能够实现几个版本的 Diffie-Hellman。

一旦完成了握手过程，数据在用户和Web服务器之间传输，在SSL记录协议中使用指定的密钥和算法。图9.40显示了记录结构。数据被划分为数据段，每一个数据段都小于16Kb。可以在每一段上选择使用压缩算法，之后，计算所选择的MAC格式，并把它添加到分段数据上。应用所选择的加密算法，添加SSL头，最后将这个包发送到网络上。

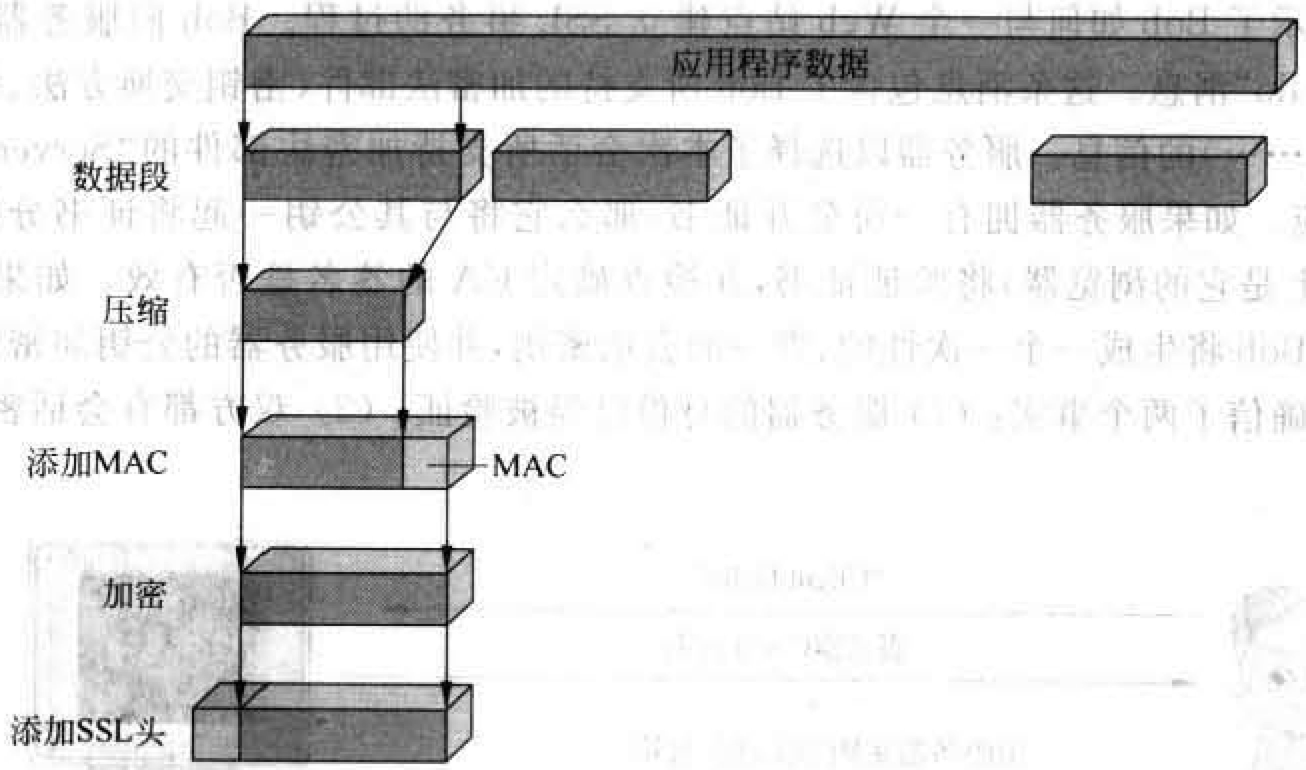


图9.40 构造SSL记录

在接收端反序处理这个过程，重新构造应用数据。你应该已经注意到，尽管这个过程看起来有点复杂，但它仅仅包含了你在这个教材中已经学习的算法和协议的应用。

9.6 历史回顾

有两个与加密法系统实现相关的事件，它们都有着历史重要性和现实重要性(实际上，事件远远不止这两个，但它们与本章的材料密切相关)。第一个事件是Pretty Good Privacy (PGP)的开发，另一个事件是正在进行中的Common Criteria的开发。两者都展示了加密法系统、政治问题、密钥管理、Internet安全、许多与计算机安全和个人隐私相关的其他问题之间复杂和相互关联的本质。

1992年，Philip Zimmermann开发了作为免费电子邮件加密程序使用的PGP。Philip Zimmermann是一位计算机安全顾问，也是一定程度的政治激进分子。他的PGP总目标是使用最好的可用加密算法；将它们集成到能够独立于任何操作系统的、易于使用的应用程序中；整个系统为最终用户免费使用。PGP的原始版本使用Zimmermann自己的块加密算法Bass-O-Matic，但很快就说明这个算法太弱了。这样他将算法替换为IDEA。他对PGP的看法并不仅仅包含一个块加密算法。现在它包含了用于密钥加密的RSA和用于散列生成的SHA-1(MD5也可以使用)。这些东西一起工作，通过一个简单的命令界面用户就可以使用它们。

Philip Zimmermann的故事也远远不仅是现在已经成为电子邮件安全事实标准的创

建。还有一个政治故事。似乎是这样，当他创建第一个版本的 PGP 时，他遇到了两个问题：RSA 被授予专利，美国政府不允许强加密法出口。他决定克服这两个问题，并把他的新程序分发给朋友。由于他的朋友都在美国国内，他并没有直接触犯美国法律，尽管 RSA 公司对他的程序感到烦恼。他最终解决了与 RSA 之间的问题，但当 PGP 出现在互联网上，因此可以被美国境外人员使用时，美国政府并不高兴。Philip 自己并没有把它放到互联网上。似乎是他的一个朋友把它粘贴到了一个 Usenet 新闻组上。粘贴的结果是 Philip Zimmermann 成为了长达 28 个月的大陪审团调查的目标。在调查过程中，Zimmermann 变成了互联网上的民间英雄。政府最终于 1996 年 1 月不加说明地终止了调查。整个事件仅仅是开放加密法与情报收集过程保护之间冲突的另一个示例。

说得简短一点，另一个重要事件是 Common Criteria 尝试的开始。现在已经很清楚，加密法和安全方面的一般问题并不仅仅是选择一个强大的算法和找到一个强密钥。一个完整的密码包可以包含多个算法、内部密钥生成方法以及用于散列和签名消息的过程（很像 PGP）。正如这个领域中一直存在的那样，问题是信任问题。你如何能够知道一个安全密码包正是你所需要的？这个问题的答案或许能够在 Common Criteria 开发努力的成果中找到。

在 20 世纪 80 年代早期，美国政府意识到需要开发一些标准，用于评测安全产品的质量。首批的这些标准之一出现在可信计算机系统评测标准（TCSEC）中，它被称为"橘皮书"。随后，与 TCSEC 有着相同目标的欧洲标准——信息系统安全评测标准（ITSEC）——也发布了。1993 年，信息安全评测通用标准（Common Criteria for Information System Evaluation）的开发努力被提高到了建立国际标准的程度。Common Criteria 为评测不仅仅限于加密法系统的安全特性提供了基础——它包含了全面的信息技术安全产品。厂商提交产品，独立的测试实验室按 Common Criteria 进行评测。评测结果为评测技术报告，在完成评测后，它为用户提供了产品能够按厂商所声称的功效工作的一种担保。

NIST 也为加密法系统提供了一个评测标准。该标准可以在 FIPS PUB 140-1 和 FIPS PUB 140-2 中找到。后一个是最新标准（实际上，NIST 已经不再使用 FIPS 140-1 评测产品了）。对于评测加密法产品来说它最重要，并且该标准与 Common Criteria 有大量的相同之处。

9.7 本章小结

Bob 和 Alice 发现迁移到更精密和复杂的加密法系统意味着他们不得不管理大量的、他们不可能记住的密钥。这样，在 Eve 破解他们的加密法可能会遇到大量困难的同时，她可能也会发现对他们的密钥造成威胁的更多成功机会。因此，他们决定使用以安全方式共享密钥的方法。他们发现的一个方法是使用诸如 Diffie-Hellman 的密钥交换协议，这个协议允许他们在可能被 Eve 窃听的通信链路上建立公用密钥。

另一个方法也出现了——认证的方法。Bob 如何确保收到的、来自 Alice 的消息未被修改，或者说它确实来自 Alice 呢？为了解决这个问题，他们找到了数种方法使用散列函数和公钥系统"签署"文档，这些签名在以后可以验证，签名的特性也能够验证消息的可靠性。

他们依然面临着另一个问题。他们如何能够确保所寻找的公钥真正属于声称属于自己

的人呢？如果他们被误导使用了伪造的公钥，那么就会对他们的消息造成危险。这就导致了他们使用证书发行机构，这个机构将会验证给定密钥的拥有人。

总的来说，所有三个人(Bob、Alice 和 Eve)都知道保护消息的安全比加密明文有着更多的内容。一个真正的安全系统必须包含管理密钥和认证消息两方面的机制。图 9.41 展示了本章给出的函数的概要。

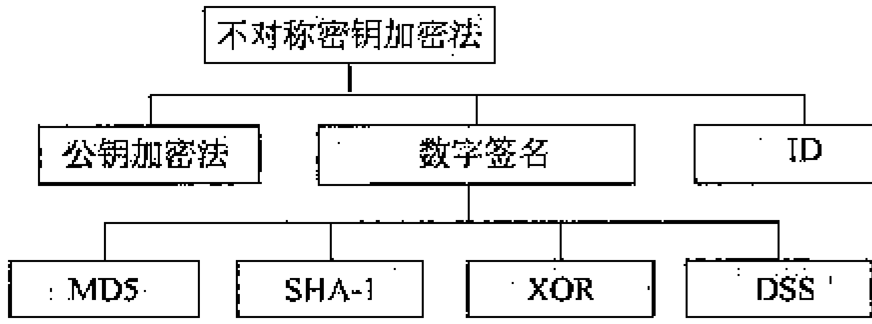


图 9.41 典型签名-散列函数

9.8 重要术语

生日问题	找到房间中有两个人生日相同的机会为 50%所需的人数。
盲签名	在不查看文档内容的情况下签署文档。
证书	将公钥值绑定到所有人身上的过程。
DSS	1994 年采用的数字签名标准。
MD5	Ron Rivest 开发的一种散列算法。
填充	散列前在消息末尾添加额外比特位的过程。
公钥基础设施(**PKI**)	一种用于建立公钥系统中用户之间信任关系的可靠和安全的系统。
SHA	NIST 开发的安全散列算法(后来更新为 SHA-1，现在是 SHA-2)。
SSL	安全套接层　用于在互联网上提供应用的安全性。
验证	验证证书依然有效的过程。

习　　题

1. 描述 9.2 节介绍的公钥授权和机密性如何满足 9.3 节列出的全部五个签名要求？

2. 9.3.1 节使用的简单 8 位块 XOR 散列函数在单词“bob”和“yes”上产生了冲突。找到使用这个函数也产生冲突的另外两个单词。

3. Alice 收到了来自 Bob 的下述消息：

“Alice it's time for us to determine a new key, Bob.”

附着在这条消息上的 16 位 XOR 签名值为 6C2E。这是一条有效的消息吗？

4. Alice 和 Bob 想要使用 Diffie-Hellman 建立一个新的保密密钥。他们的公钥值为：

$$p = 26586842533836180377$$

和

$$g = 155801347$$

Alice 向 Bob 发送了下述值：10029496837244214838。

Bob 的私有数字是 2667001284。

Bob 向 Alice 发送什么？密钥值是什么？

5. Alice 向 Bob 发送了下述消息：

"Bob, let's meet at the usual place at noon tomorrow, Alice."

她使用 MD5 散列消息，之后使用 16 位块长度的 RSA 加密散列值。Alice 的 RSA 公钥为：

$$N = 1896593811505640680937731503 91$$

$$d = 1175885676400129$$

下述哪一个加密散列值是消息的正确散列值？正确的散列值与不正确的散列值之间存在的差别在什么地方（也就是说，有多少个字符不同，位置在什么地方）？

(a)　47951043611207227029563446581
352888618914934857015185O3049
54062023219871197095328143119
169788180095885361297024500881
742511236195995243479936942 99
36954262374929321715217941245
11449857985423730288736059989
13430732301535895640003706330
15538387441415917005048020880
163809900844908575360903122072
144960596826920650018699165328
95163607844779107516456274369
103918969368387484447914139704
76069742877758574656041687 77
152749496009907655063958072802
45748928547946983699201341618

(b)　47951043611207227029563446581
184161882656245880321888137658
900682640911164615329925220915
49869178049443931329254150168
74251123619599524347993694299
163033944196110604975979245321
81082243942335608898626549219
140346124094786061957926316143
65569882893207348447027812642
99525581119696307868751903848
74251123619599524347993694299
81082243942335608898626549219
103918969368387484447914139704
76069742877758574656041687 77
152749496009907655063958072802
45748928547946983699201341618

6. Bob 想向 Alice 发送一条由第三方签署的重要消息，但他并不想让第三方知道消息的内容。它使用随机数 100372401233791 隐藏消息，并把它交给了 Trusted Notary Public Service。使用 16 位 RSA，该公司签署了消息，并把它返回给 Bob。盲签名消息为：

7588316241626312733541820l663
45961554160994101449757034 21
158728952869916178458012134745
118526025261370562318179750342
158728952869916178458012134745
182130930859830580308433590877
85592668333782584212284483527
162181093993910088901552861810
157303387376692959649495686256
120472171820911393013720567791
50091629346356337188546443383
17940439025268923865108585938
158728952869916178458012134745
16568420109629692448130283449
33002437160194189445292159808
118526025261370562318179750342
158728952869916178458012134745
80157452348797548678511728170
45961554160994101449757034 21
38399746163487814429807993790
180557696831286742925905739086
162724727974146105457778034593
100906556339429695230686193604
88439305486904302466411038994
164285278023193840937653051650
72106808602161891422961402423
94539437763177570338591276765

(a) Bob 发送给 Alice 的签名消息是什么?

(b) Trusted Notary Public Service 的公钥为:

$$n = 189659381150564068093773150391$$

$$e = 53874618627671702415711710977$$

Bob 发送给 Alice 的实际消息是什么?

7. 建议一种可以使用 XOR 散列生成随机数的方法。

8. 使用 SHA-1 随机数生成器生成一组五个随机数。密钥为 100003461。

(a) 这五个随机数是什么?

(b) 将它们转换为二进制,并对结果运行随机数测试。它真的是随机的吗? 为什么是或为什么不是?

9. Bob 向 Alice 发送消息:"Alice I agree to meet with you on Sunday at noon Yours, Bob"。他使用 8 位 XOR 散列签署了消息。

(a) 散列值是什么?

(b) Eve 截获了消息,并改变了一个重要细节,但她的消息依然保持散列值不变。Eve 会给 Bob 发送什么消息呢?

10. 使用经典加密法也能够创建散列值。例如,使用 Autokey 加密法的一个版本:

(a) 选择一个长度为 x 的关键词。

(b) 利用 Vigenere 加密法、使用关键词加密明文的前 x 个字符。

(c) 使用密文字符作为下一块 x 个字符的关键词。

(d) 重复这个过程(以字符串 y 的形式在明文末尾添加填充字符,确保最后一块的长度为 x)。

(e) 散列值为最后一块的密文。

使用这种带关键词的方法,写出如何找到下述明文的散列值。

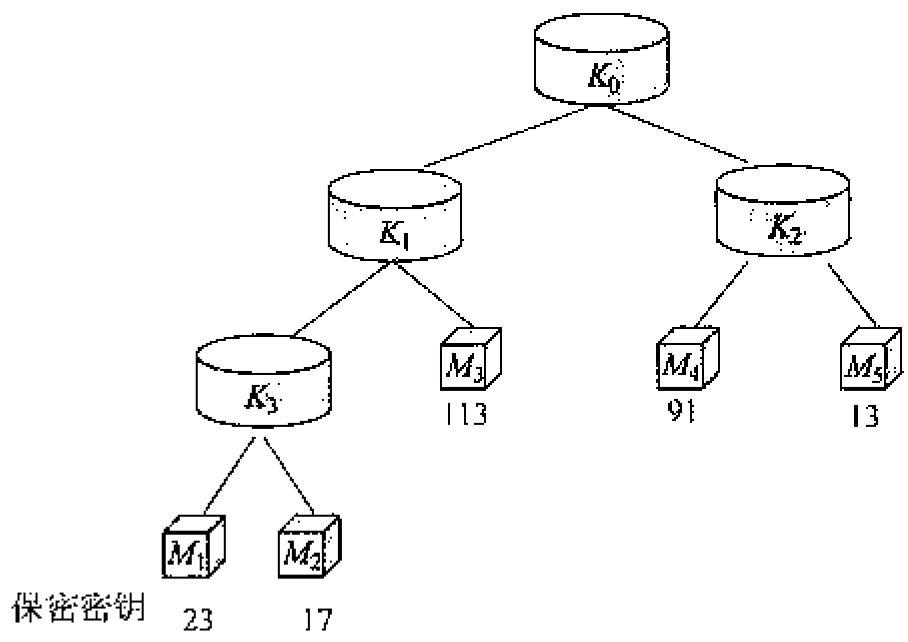
"Alice,this is a test of a new hashing method. Bob"

如果将明文修改为下述文字,散列值会如何变化?

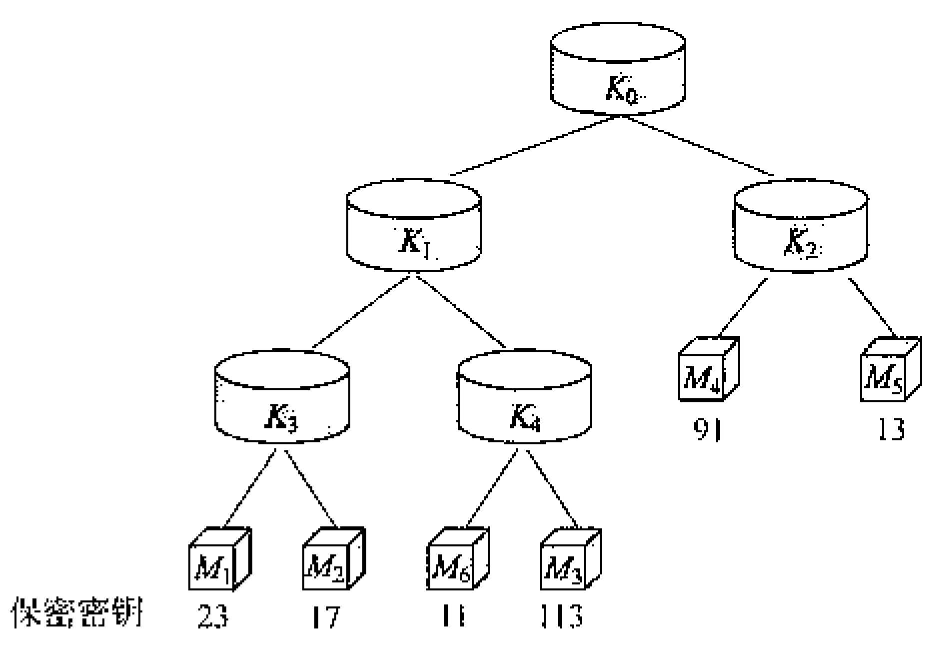
"Alice,this is the text of a new hashing method."

11. Alice 和 Bob 使用授权的 Diffie-Hellman 方法建立密钥,这里,$p=3061$,$g=17$。Alice 的保密数字是 113,Bob 的保密数字是 211。Alice 选择随机数 91。Bob 选择随机数 11。Alice 向 Bob 发送了什么? Bob 向 Alice 发送了什么? 他们最后的密钥值是什么?

12. 给定下述组密钥结构,$p=3061$,$g=13$,找到每个节点的 K 和 BK。



13. 一位新成员想要加入到问题 12 结构的组。其结果为下述结构:



给出重新生成这个系统密钥的结果。

14. 开发一种新的 Web 安全协议并使它得到全世界的接受。在老师指导下为其命名。

第三部分　密码学的未来

第一部分介绍了密码学的历史基础知识。在这个时期中，智慧和毅力起着主导作用。第二部分介绍了密码学的现状，其中计算机起着主导作用，当然智慧和毅力对开发人员来说仍同样重要。

在第三部分中，我们将进入一个全新的世界，以前的知识不再适用了。这是一个充满幻想的世界；它需要掌握全新的智慧和毅力；它使密码学发生了翻天覆地的变化，重新改写了密码、加密法及其密码分析的本质。这就是量子物理的世界。

第 10 章　量子密码学

10.0 概　　述

从使用简单替换的恺撒加密法，到诸如 AES 之类的只能用计算机才能完成的复杂替换的置换加密法，密码学走过了一个漫长的道路。看起来未来能做的也大概就这么多了。但这是完全不真实的。我们现在处在了密码学(和计算机)的另一个变革边缘，它将比过去 50 年的发展更大。计算机将进入量子力学的世界，关于现实世界的一些运算思想将发生根本的变化。一旦进入了这种新的世界，密码学将会发生什么变化，我们现在也只能是猜测而已。

这就是 Bob、Alice 和 Eve 将要进入的世界，正如 Bette Davis 曾经说的："请系好安全带，我们要发飚了。"

10.1 量子系统概述

早在 1965 年，也就是在硅片上设计电路的最初技术开发之后，戈登・摩尔，Intel 公司的创始人之一，发表了著名的"摩尔定律"，该定律用现在的话就是，硅片模板上晶体管的数量每 18 个月将翻一番(他的最初预言是每年翻一番)。这种发展趋势在未来十年还将会继续。但是，摩尔定律最终将走向死胡同。随着晶体管变得越来越小，它们所含的硅原子也越来越少。最终，每个设备都将只含有(或作用于)少量的电子。此时，电流变得不稳定，设备的行为也不再是可控的了。而且，如果晶体管薄到只有 6 个量子厚度时，其绝缘性将不存在了。摩尔定律同样适用于光纤通信。当前，大约用一万个光子表示一个位，但随着通信变得越加稠密，所需要的光子越来越少。最后，将减少到用一个光子表示一个位，此时，我们用来评估位的标准将不再适用了。为了不断获得更高的计算能力，必须开发新的数字电路设计方法。现在正在开发两种方法：DNA 的使用和量子计算。其中，量子计算看来对密码学的影响最大。

10.1.1 量子位

在经典(即非量子)世界中，是用电压来表示二进制位的。晶体管用来逻辑操作电压，实现布尔函数。在量子世界中，二进制位用量子位(qubit)表示。一个量子位在量子系统中有两种状态。它是具有**上旋**(spin-up)和**下旋**(spin-down)的电子，或是可以在某个方向上发生偏振的光子(光的粒子)。例如，可以用处于上旋状态的电子表示二进制位 0，处于下旋状态的电子表示 1，如图 10.1 所示。量子的状态用 $|0\rangle$ 和 $|1\rangle$ 表示，称为**刃**(ket)(这种表示法

称为迪拉克表示法，它用括号和竖线以$\langle a|b\rangle$的形式来定义量子状态。)。

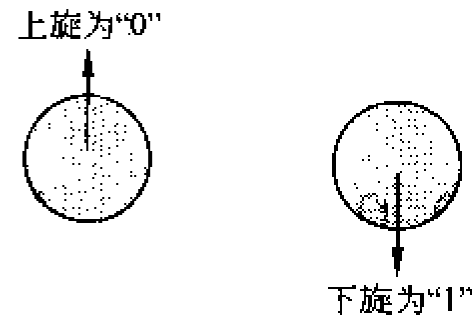


图 10.1　量子位的状态

正如在经典世界中的那样，如果要构建量子计算机，就需要有用来度量量子位的值以及逻辑操作量子位的方法。这还只是困难的开始，在量子世界中，度量方法发生了变化；操作位的方法也与以前常用的有所冲突。为了帮助读者理解量子位的这些非常属性，有必要介绍一下量子物理的一些特性。

10.1.2　量子物理的世界

早在20世纪，物理学家就开始探索光、能量和粒子的行为，它们以一种以前从未了解的方式组成了原子。在这个过程中，他们打开了潘多拉盒子，魔鬼理论和不可接受(在当时来说)的预言出来了。所有的规律都发生了变化，所有事情的行为都不像以前那样了。这是始于德国物理学家 Max Planck 的，他发现能量是以固定大小的束进行运动的，称之为**量子**(quanta)。1905年，爱因斯坦预言，光也是以固定大小的束传播的。到了20世纪20年代中期，我们认为已经了解的所有事情都发生着变化。但这其实还只是刚刚开始。

解释这个陌生世界的最著名且依然令人费解的是 Young 的**双缝**(two-slit)实验。该实验的结果使波与粒子的差别模糊了。该实验首先用一个光源穿过有两个缝的墙面，目的是要观察光在第二个墙面上的相互干扰方式，如图 10.2 所示。光在穿过这两个缝后，会发生自干扰，在第二个墙面上生成一些光线和暗线。

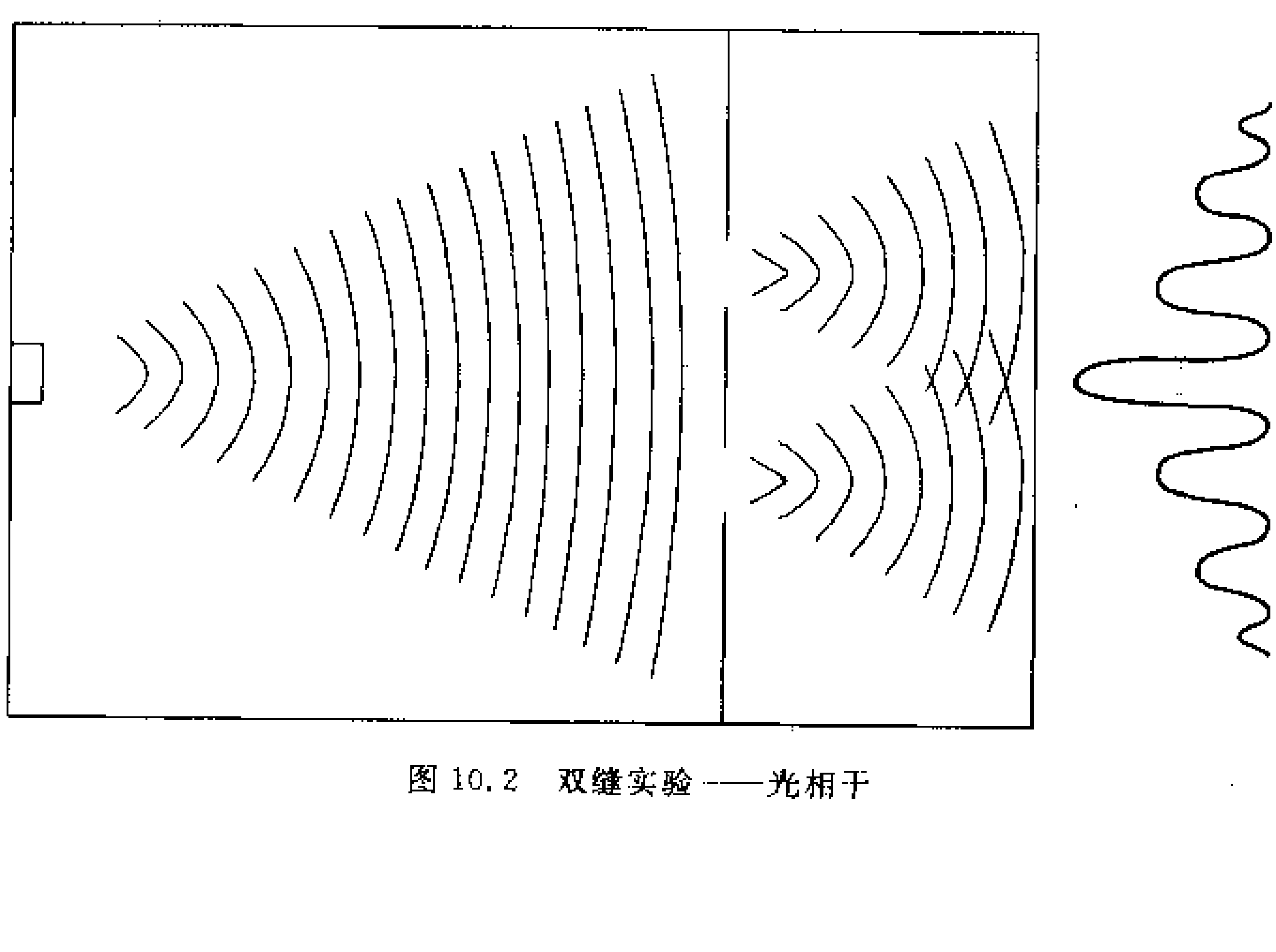


图 10.2　双缝实验——光相干

这并不奇怪，毕竟，光是一种波，是可以穿过缝的。真正令人惊奇的是当用电子而不是光波来进行该实验时，当电子穿过两个靠得很近的缝后射在第二个墙面上时，人们原以为电子撞击第二个墙面的位置应取决于它所穿过的缝。但是，在量子世界中，事情并不像通常那样。使用电子的双缝实验如图 10.3 所示。显然，电子会相互干扰，但每次只有一个电子发出。因此，真实的情况是只有一个电子发生了自干扰，这说明该电子是同时穿过这两个缝的。当我们进一步观察这个电子穿过这两个缝的情况时就更加令人惊奇了。如图 10.4 所示，如果在两个墙面之间放置一个传感器，这种自干扰就会消失，即该电子只会穿过其中的一个缝。看来，当我们对它进行观察时，该电子改变了其行为。这个有着百万奖金的问题是："在该电子穿过缝之前，它是如何知道有人在观察它，从而改变了其行为的？"

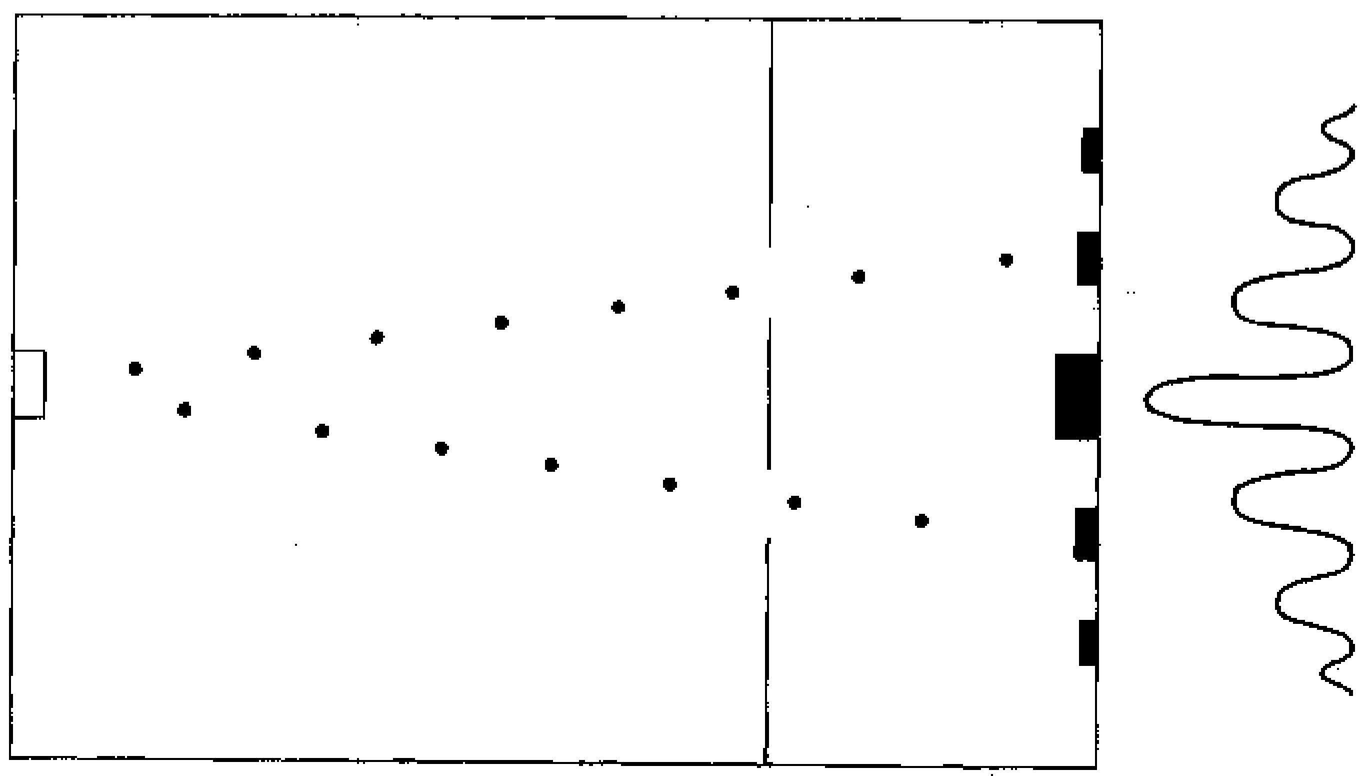


图 10.3 双缝实验——电子相干

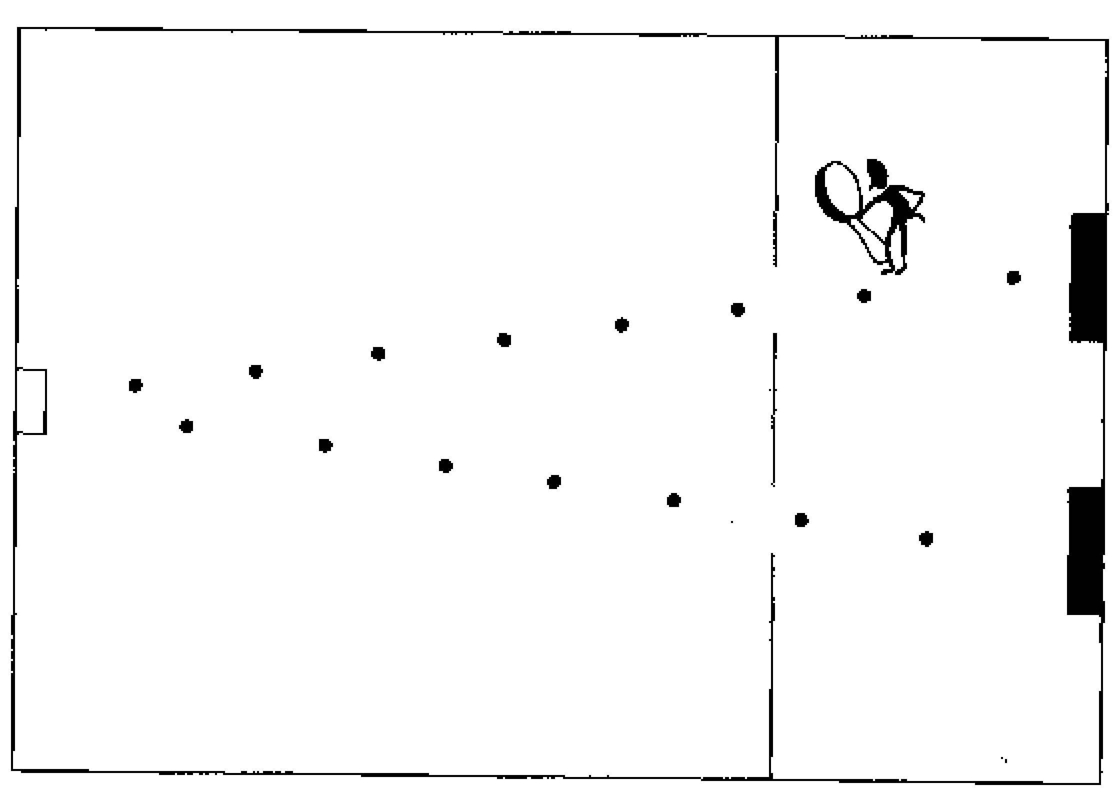


图 10.4 双缝实验——观测电子

答案是在我们观察这个量子系统时会影响该系统的行为。传统的观念认为，我们作为外部观察者进行实验时，不会影响实验的结果，但这在量子世界中不再是正确的了。这可能会引起实验物理学家的一些关注，但对密码学家来说，所关注的是最终入侵检测系统（后面将会详细介绍）。

量子系统还有四个令人惊奇的特征，很值得关注。第一个是**叠加**(superposition)原则，即是，量子有多种状态，且它可以同时处于所有状态的混合态。这是量子与二进制位的根本区别。二进制位只限于0或1两个值，而一个量子位则可以为|0>、|1>或这两者的组合（用$a|0>+b|1>$表示）。换句话说，量子位可以同时为"0"和"1"！该特性使得量子计算机的运行速度惊人地快（后面将详细介绍）。

量子系统第二个令人惊奇的特性称为**纠缠性**(entanglement)。它表示，如果两个量子粒相互作用，那么它们可以从这种相互作用中出现某种"连结的"状态。假如它们分离一段距离后，它们仍可以即时地相互"通信"。如果其中一个发生变化，另一个也会发生变化，这可以使信息传输的速度比光速还快。纠缠理论是量子远程移动通信(teleportation)研究的基础。你不要以为这还只是理论，已经得到实验验证了。这意味着两个量子可以成为纠缠状态，从而可用来进行二进制信息的远程通信。我们真的是生活在一个令人惊奇而陌生的世界中。

量子系统第三个令人惊奇的特性是**不可复制定理**(no-cloning theorem)，它表明，不可能生成一个未知量子状态的完整副本。第四个是**测不准原理**(uncertainty principle)，它表明，不可能完全知道量子系统的物理特征。对一种特征的度量将会改变另一种特征。

10.2　量子因子分解

量子计算的概念已经探讨有好长一段时间了，但在成为一门严肃的研究主题之前，必须有人先提出一个实际应用。1994年，当Peter Shor在贝尔实验室工作时就完成了该工作。他发明了一种能高速地将大整数因子分解的量子算法，称为Shor算法。该算法是基于称为"用查找秩来因子分解"的经典因子分解法。显然，该算法对密码学有很大的影响。如果能实现一个高速的量子因子分解系统，那么诸如RSA和椭圆曲线之类的加密法就要求有更大的密钥。

事实证明，整数N因子分解问题就等价于数列$x^0 \pmod N), x^1 \pmod N), x^2 \pmod N)\cdots$，的周期$r$，其中$x$是与$N$互质的整数（即$x$与$N$没有公因子（1除外））。该周期$r$是使$x^r=1 \pmod N)$的最小整数，称之为$x \bmod N$的秩。当$r$不是$N$的分解因子时，就可用它来计算$N$的分解因子，这可用$\mathrm{GCD}(x^{r/2}+1,N)$和$\mathrm{GCD}(x^{r/2}-1,N)$表示。这其实只是对Fermat因子分解算法（详细介绍见第8章）的一种数学修改。

例如，用$x=23$来将143因子分解，生成以下数列：

1，23，100，12，133，56，1，23，100，12，133，56，1，…

该数列的周期为6，因子143的分解因子可表示为：

GCD(23^3+1,143)

和 GCD(23^3-1,143)

其中 GCD(12168,143)=13 和 GCD(12166,143)=11。

CAP 软件实现了该算法(记住,它不是量子算法,而是量子算法的基础)。在 CAP 的主菜单中选择 Integer 项,打开 Long Integer Routines 窗口。选择 Special Functions 下的 Factoring-Shor's Algorithm 项,如图 10.5 所示,输入一个 N 值,再输入一个 x 值或由 CAP 软件选取一个值。单击 Run 即可计算 N 的分解因子(注意,该数列显示在 Results 窗口中)。

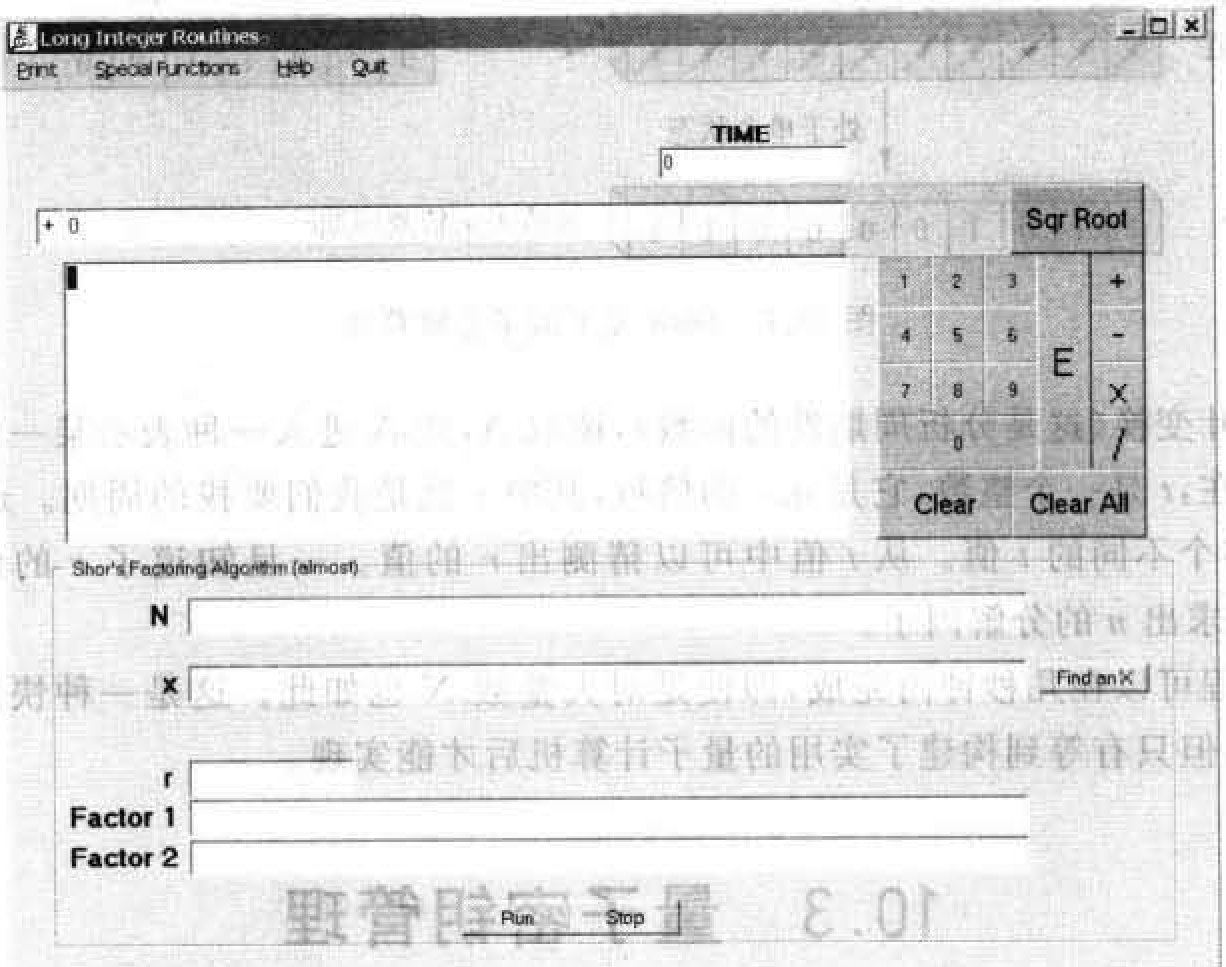


图 10.5　Long Integer Routine 窗口

Peter Shor 将该分解因子算法进行了改建以充分利用量子计算的两个特性:纠缠性和叠加性。该算法的详细内容涉及了一些数学问题,它们超出了本书的范围;但是,用该算法实现的量子系统可以用一种准确但简单的方法来解释。

假设要分解因子的数字为 n,找出一个整数 q,它是 2 的幂且位于 $n^2\sim2n^2$ 之间。选取一个与 n 互质的随机整数 x。这些可在传统的计算机中完成。现在来生成两个量子寄存器 A 和 B,使 A 足够存储 $q-1$,B 足够存储 $n-1$。如图 10.6 所示,设寄存器可为某值 y,寄存器 B 为 0。由于 A 是量子寄存器,除非其内容不确实,否则,根据叠加原则,它可以包含 y 的所有可能值。在寄存器 A 中应用函数 $x^y \bmod n$,并将结果存储到寄存器 B 中。如果 B 的内容未确定,那么 B 也是叠加状态,可以包含所有可能的函数值(y 的每个可能值)。该结果是在一步中含有大量的并行计算。另外,这两个寄存器现在是纠缠的了,因此,一个寄存器的变化将影响另一个寄存器的值。

利用 A 和 B 的纠缠性,读取寄存器 B,将使 B 从叠加状态中出来,进入表示单个值(假设为 R)的状态。由于 A 与 B 是纠缠的,它含有所有 y 值的叠加;使 $x^y \bmod n=R$,对 A 进

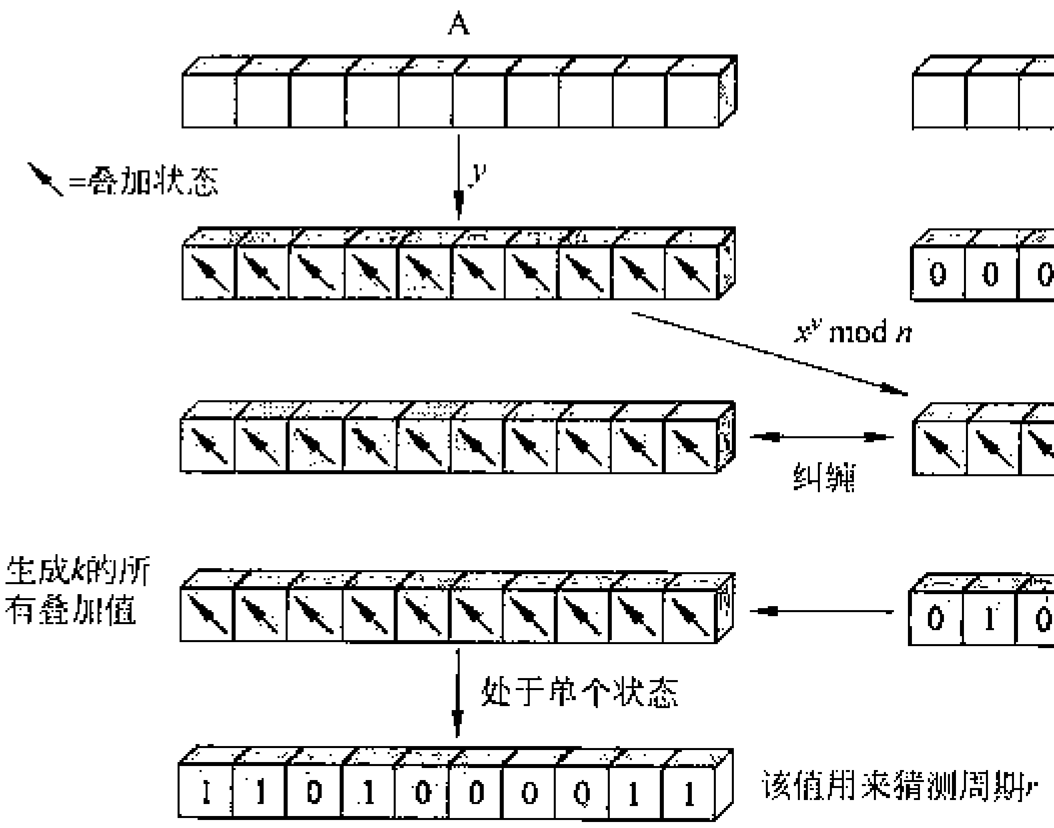


图 10.6 Shor 量子因子分解算法

行离散傅里叶变换(这是分析周期性的函数),读取 A,使 A 进入一种表示某一值(假设为 t)的状态。现在,t 是一个整数,它是 q/r 的倍数,其中 r 就是我们要找的周期。运行该算法 n 次,以得到多个不同的 t 值。从 t 值中可以猜测出 r 的值。一旦知道了 r 的值,就可以用 GCD 计算来求出 n 的分解因子。

整个过程可以在几秒钟内完成,即使是对大整数 N 也如此。这是一种快速、高效的因子分解算法,但只有等到构建了实用的量子计算机后才能实现。

10.3 量子密钥管理

对密码学来说,另一个应用是利用其不可复制定理和测不准原理,来确保双方(假设是 Alice 和 Bob)之间保密密钥的安全传输。Bob 和 Alice 面临的一个问题在第 9 章介绍过。他们如何才能确定一个保密密钥仍是可靠的,Eve 没有发现或修改过?第 9 章中介绍的解决方法是使用公钥系统、数字签名、认证或密钥交钥算法,如 Diffie-Hellman 过程。量子系统提供了另外一种高度安全的选择。

量子密钥管理最常用的方法称为 BB84 协议(以 Bennett 和 Brassard 的名字命名,他们于 1984 年发表了关于该过程的论文)。该方法使用的是光子(光的粒子),因此它在光纤链路中很容易实现。它将二进制位 0 和 1 按光子的偏振(电场的方向)加密。光子可以在水平、垂直或对角线(+45°和−45°)发生偏振,如图 10.7 所示。

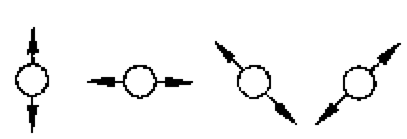


图 10.7 偏振光子

这里要构建两个过滤器，一个用于水平和垂直偏振的光子，另一个用于斜线偏振的光子。如果光子穿过一个与它的偏振方向匹配的过滤器，不会发生变化。但是，如果穿过的是不匹配的过滤器，那么它将随机地变为该过滤器的某个偏振方向，如图 10.8 所示。这里要记住的关键点是这种改变是随机的。

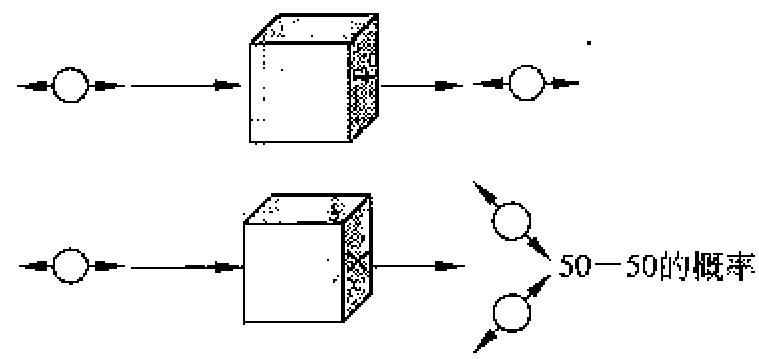
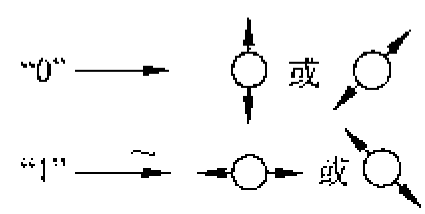


图 10.8　偏振光子通过过滤器

由于 Alice 和 Bob 的目标是要达成形成互信密钥的位列，他们首先要决定哪个偏振方向要赋给哪个位值。例如，他们可能决定用“0”来表示垂直和 45°方向偏振的电子，那么“1”就表示水平和－45°方向偏振的电子。



接下来 Alice 选取一个可能的密钥位，发送一个随机偏振方向的光子给 Bob，她记录下这个偏振方向。Bob 接收到这个光子后，随机选择一个偏振过滤器。他记录下过滤器的类型和结果位值，如图 10.9 所示。当 Alice 发送完所有的光子后，在一个非安全的信道（也许是电话）上与 Bob 联系。Bob 将每位所用的过滤器类型告诉 Alice，Alice 则告诉 Bob 他选择的哪些过滤器是正确的。Bob 选择了正确过滤器的那些位就形成了互相信任的密钥。

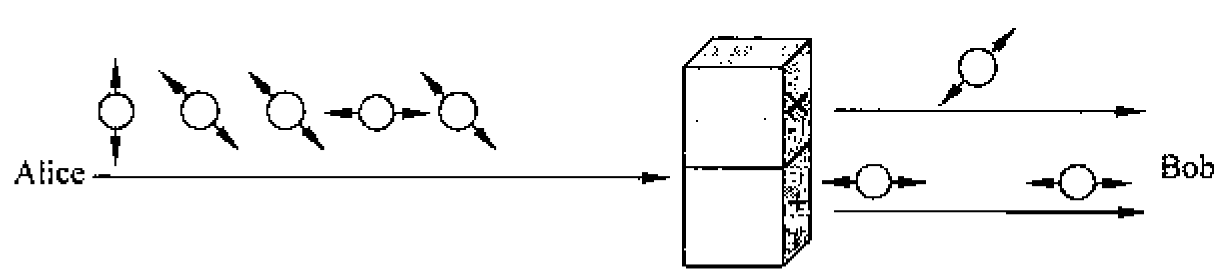


图 10.9　Alice 和 Bob 确定保密密码

例如，Alice 生成了 10 个随机位：1101000100 和 10 个偏振方式。如图 10.10 所示，Alice 使用－45°偏振方向的过滤器发送第一个位（“1”），使用垂直偏振方向的过滤器发送第二个位（也是“1”）。在接收端，Bob 按以下顺序选择偏振过滤器（x 为斜线，＋、－为水平、垂直方向）：$+xx++xxx+x$，因此他得到位 1001010101。当他与 Alice 打电话后，他们就发现，Bob 在 3、4、5、7、8、9 位上的选择是对的，这样他们俩就知道密钥为 010010 了。

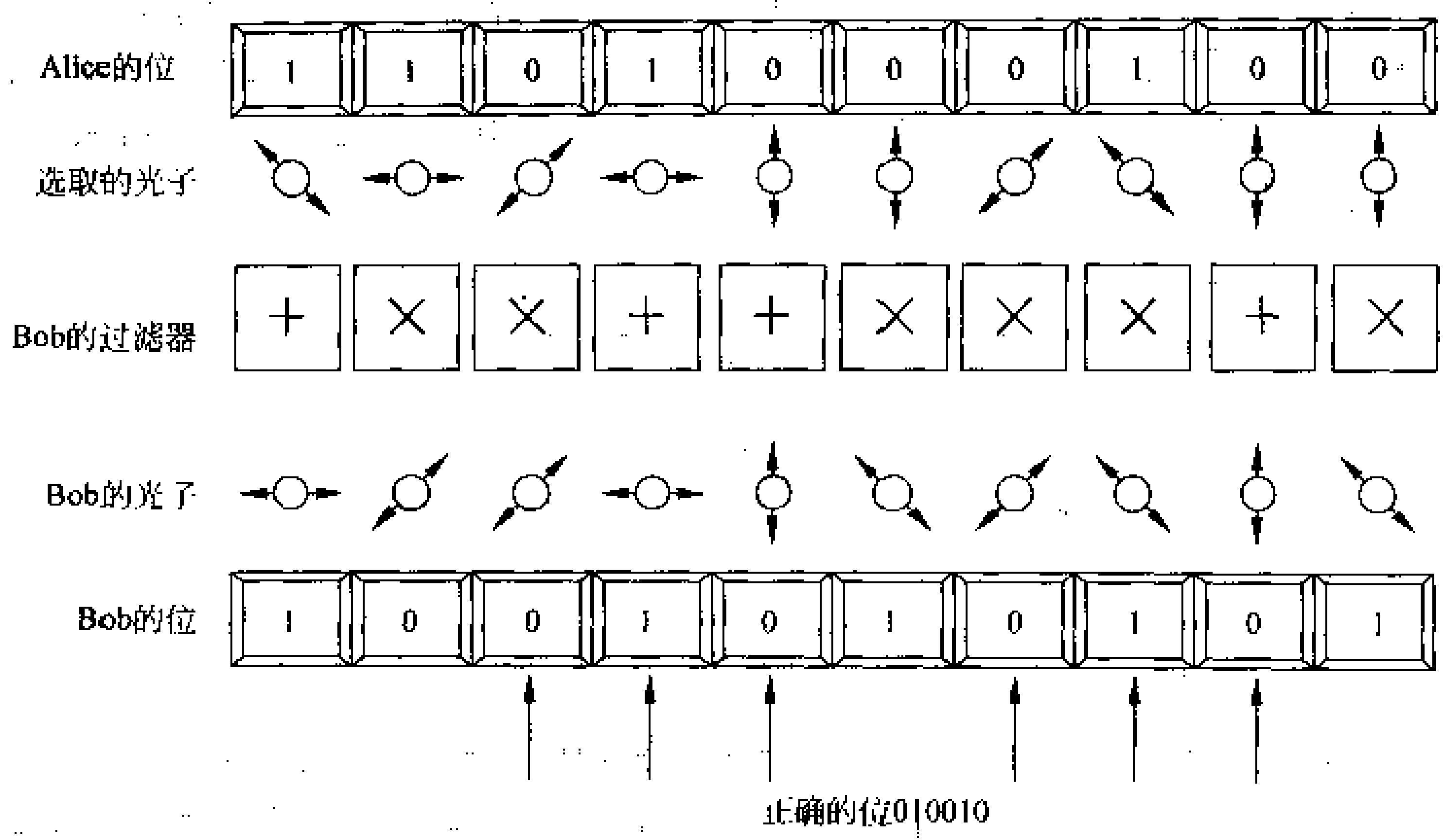


图 10.10 确定密钥的位

10.3.1 窃听

如果 Eve 不去干预这个过程，那么 Alice 和 Bob 就可以有保密的密钥了。但是，如果 Eve 进行窃听，并且像 Bob 那样随机选取过滤器以生成位，那将会如何呢？如果她偷听了 Alice 与 Bob 的对话，她能有发现密钥所需的信息吗？答案是没有，原因是，假设 Eve 截获了他们发送的光子，如图 10.10 所示，像 Bob 一样读取这些光子，然后将它们发送给 Bob。整个过程如图 10.11 所示。

当然，Eve 不知道她的过滤器选择哪些是正确的，她也不可能打电话给 Alice 来确认她的选择。但是，她可以偷听 Alice 与 Bob 之间的谈话。在上面示例中，她知道正确的过滤器是 $x+x+++xx++$。Bob 在 3、4、5、7、8 和 9 位的选择是正确的。Eve 知道她在 2、3、8 和 10 位的选择是正确的。因为 Bob 和 Alice 只使用由 Bob 标识的正确位，因此 Eve 只知道密钥的两个位(即第 3 位和第 8 位)。

不仅 Eve 无法确定正确的密钥，Bob 和 Alice 也可以检测出 Eve 在这个过程中的干预。注意，在上面示例中，Bob 认为密钥是 010011，而 Alice 知道的密钥位为 010010。尽管 Bob 使用的过滤器是正确的，但最后一位是不同的。之所以不同，原因在于 Eve 使用了错误的过滤器，因此，当 Bob 使用正确的过滤器时，他有 50%对 50%的机会还原正确的位或生成错误的位。在这个示例中，他使用正确的过滤器生成了错误的位。如果 Alice 选取密钥的一个小子集，并将它告知了 Bob，Bob 和 Alice 就可以检测出这种改变。如果第 9 位是子集的一部分，那么 Bob 就告诉 Alice。他们也就都知道有人截获所发送的光子了，因此，他们取消了该密钥并重新换用其他密钥进行尝试。这个过程称为**密钥调整**(key reconciliation)。

只要阅读通过公共信道的小子集的位，尤其是子集中没有不正确的位，Eve 就可以获得

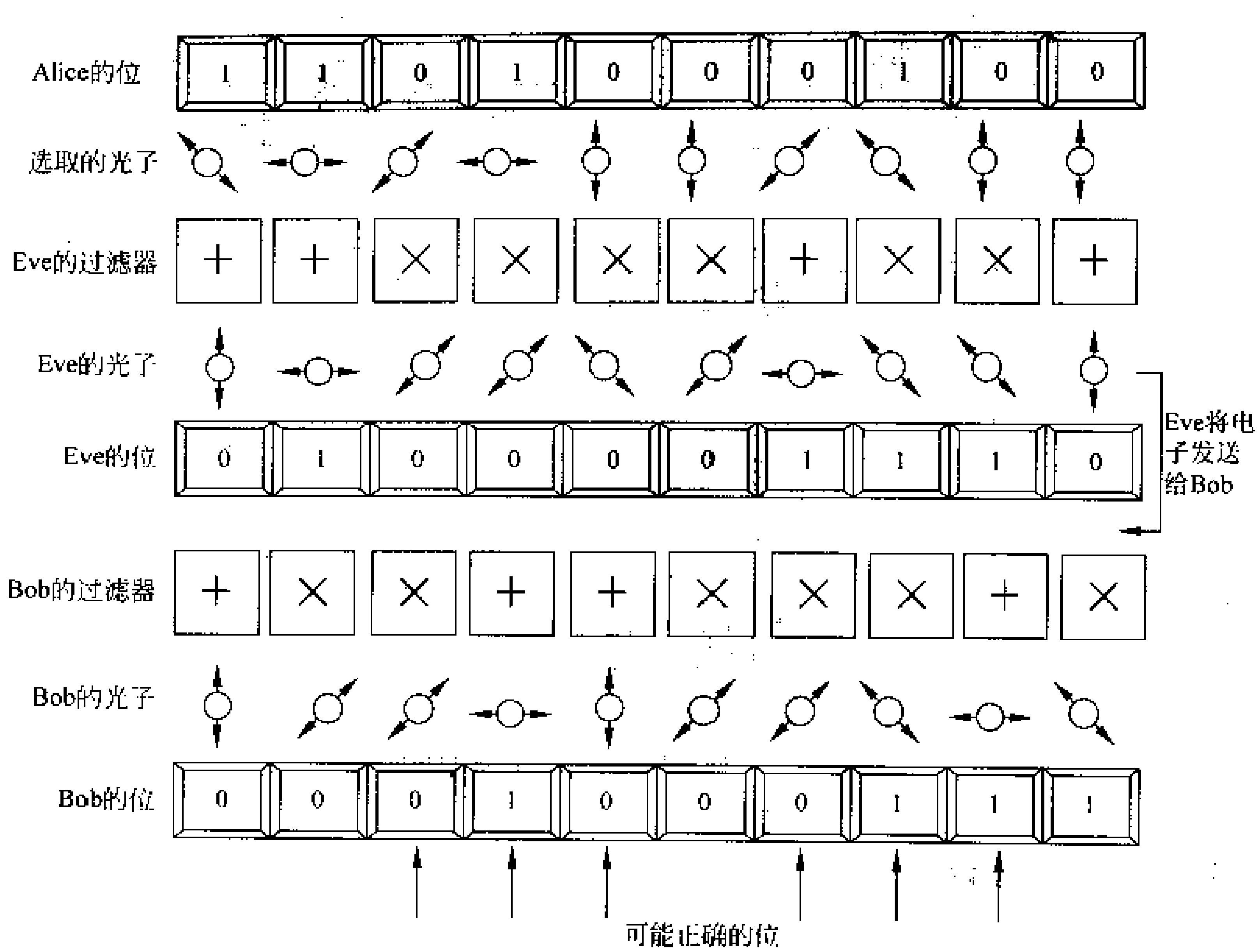


图 10.11 Eve 干扰后的结果

额外的一些信息。另一种方法是，Alice 和 Bob 就密钥位的随机小子集进行确认，然后只比较这些子集的校验位（也就是说，计算该集中 1 的个数，如果为奇数，那么该集的校验位为偶校验；否则，为奇校验）例如，如果 Alice 和 Bob 确认了保密密钥的 20 位，但 Eve 截获了这个密钥交换，在 Bob 的密钥中有未检测的错误，如图 10.12 所示。由于 Eve 的干预，Alice 和 Bob 未能确认 2、5、8、13 和 18 位。为了发现问题，Alice 和 Bob 决定运行一系列的奇偶校验。在公共信道上，他们选取 1、3、9 和 15 作为第一个测试集。在未告知具体位的情况下，他们都返回了一个奇校验，因此目前看来没发现什么问题。他们的第二个测试集使用 2、5、14 和 19 位。这次他们都确认了偶校验，尽管 Bob 的位与 Alice 的有所不同——他们遗漏了真正的错误。但是，对于第三个测试集 4、8、10 和 20 位，Alice 得到的是奇校验，而 Bob 得到的是偶校验。这样，就可以发现 Eve 的干预了，Alice 和 Bob 都不需要去找出究竟是密钥的哪个位有问题。

Eve 可能会试图这样来避免这个问题：复制 Alice 的光子，度量其中一个，将另一个发送给 Bob，这样 Bob 收到的就是正确的光子。量子系统的不可复制性可以防止 Eve 这样做。如果不知道 Alice 的光子的完整知识，Eve 无法复制。要获得完整的知识，Eve 就必须全部度量它们。而度量它们将使其中的某些发生变化，因此，她又回到了上面所介绍的问题中，Alice 和 Bob 总是可以检测这种干扰的。

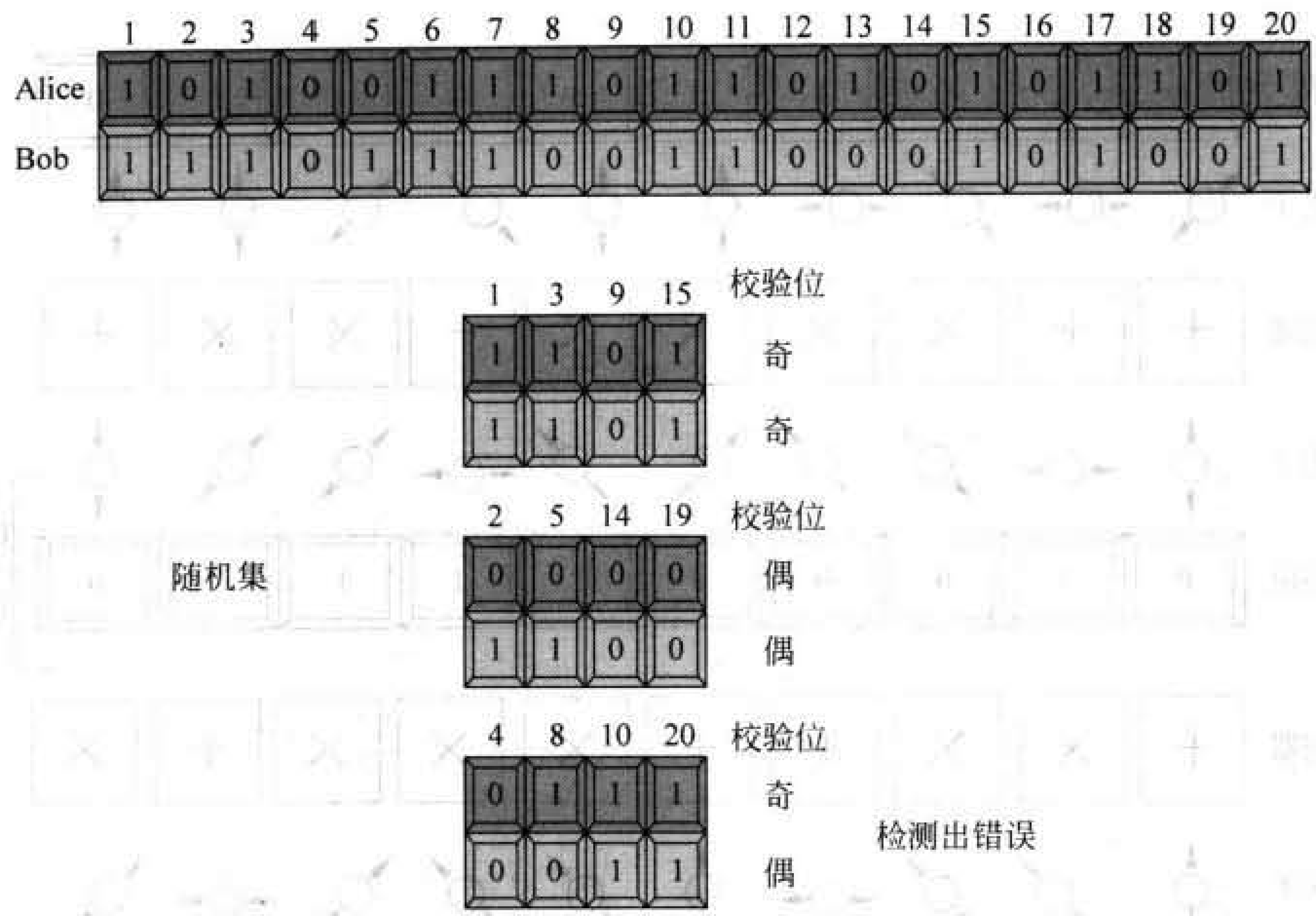


图 10.12 使用校验位进行密钥调整

10.3.2 实验验证

量子密钥发布(Quantum key distribution,QKD)不只是在理论上可行,已经在实验室得到了实现。Los Alamos 国家实验室在 48km 的光纤上演示了一个 QKD 过程。在日内瓦大学,已经构建了一个 QKD 实验,光纤的距离达 70km,速率为 100Hz。所有这些实验说明,QKD 是可行的,但还需要进一步的工程设计,使它可以应用于实际的使用中。

10.4 本章小结

从最初的用一个字母替换另一个字母开始,密码学已经走过了一个漫长的过程。在密码学发展的每个阶段,都出现了新的挑战,对开发人员和用户要求有更大的创造性和毅力。字母出现频率分析可以破解大多数的替换加密法,促使了多码和多图加密法系统的开发。密文模式的发现以及已知单词的使用,使得这些加密法也是不安全的。变换和变换—替换的组合使用,也曾一度提供了加密法的安全性,但是,计算机密码分析的开发,使得这些加密法也变得不安全。今天,计算机既已应用于密码分析,也已应用于加密法的实现了。计算机技术的发展,使得人们开发了非常复杂的替换—置换模式,以及基于非常难以求解的数学问题的加密法。寻找加密法的弱点与开发更强大的加密法之间的斗争还将继续。未来,这种斗争将进入一个新的领域——量子物理世界。

这不是密码学故事的结束,而只是刚刚开始。只要我们需要维护我们的隐私,只要有秘密,只要有人贪婪,我们就需要隐藏信息。在上面进行的 Alice、Bob 与 Eve 的战斗中,将会有另外一个参与者——Paul(或 Paula),他将作为数据的保护者——译解密码者。Paul 的

工作是确保 Alice 和 Bob 有最安全的方法来保护他们的通信，这要求 Paul 应知道 Eve 可能破解 Alice 和 Bob 安全性的所有努力。在一定程度上来说，本书的目的就是帮助你能胜任 Paul 的工作。如果是这样的话，那么我祝你好运！

10.5 重要术语

BB84	一种量子密钥发布协议，由 Bennett 和 Brassard 于 1984 年开发的。
纠缠性	两个量子位可以远距离共享状态的能力。
密钥调整	将密钥位进行比较的一种过程，以检测错误或第三方的干预。
量子位	在量子系统中与二进制位的等价体。
Shor 算法	是量子计算最早的实际应用之一，是一种高速的、可对大整数进行因子分解的算法。
叠加性	在量子系统中，量子可以同时处于所有可能状态的一种能力。

习　题

1. 使用 CAP 软件中的 Shor 算法将下列数字因子分解，并用图形表示出数字的位数与因子分解所用时间的关系。从中你可以得出什么结论？

(a) 481

(b) 4169

(c) 154811

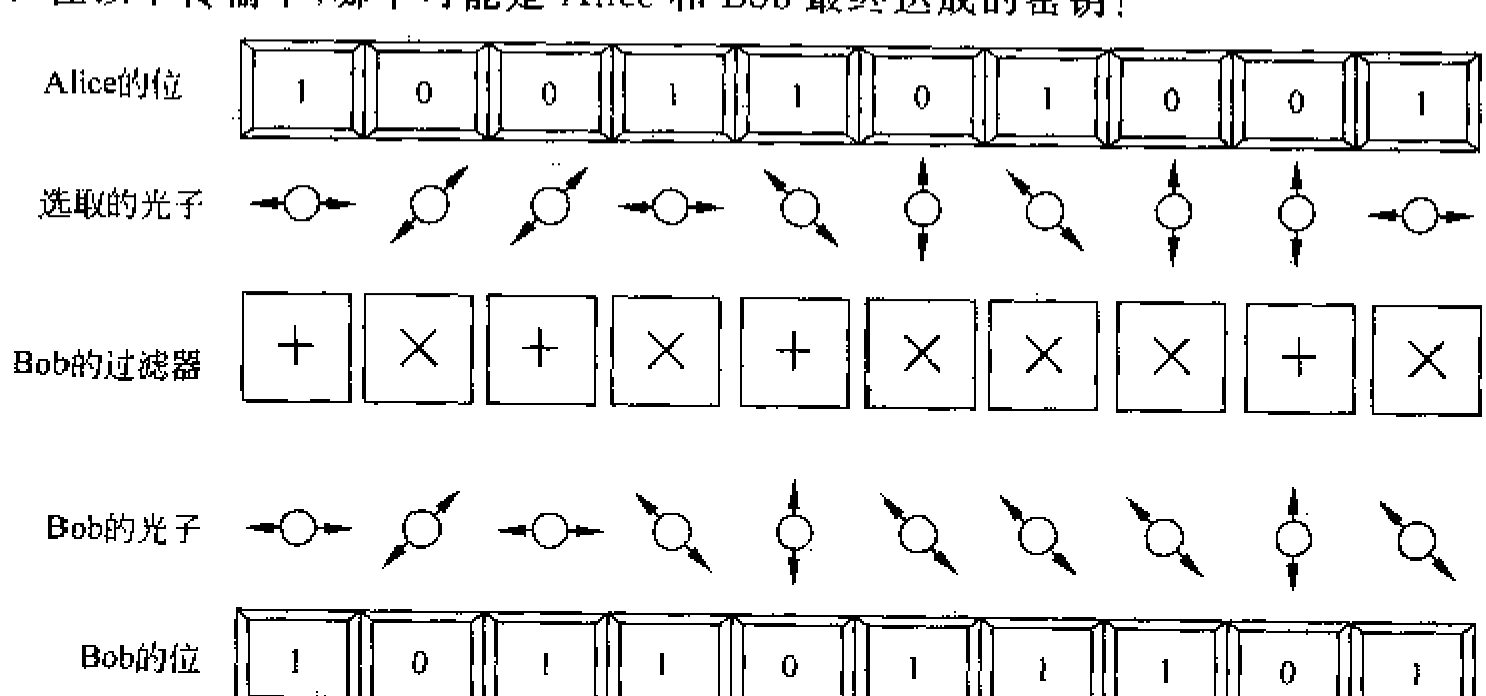
(d) 254497

(e) 1096409

(f) 16135487

(g) 178443493

2. 在以下传输中，哪个可能是 Alice 和 Bob 最终达成的密钥？



3. Bob 和 Alice 达成了以下位：

Alice 1 0 0 1 0 0 0 1 1 1 0 1

Bob 1 0 0 1 1 0 0 1 1 1 0 1

其中有一个错误。如果先将它们分成大小为 4 的密钥子集，请给出定位并修正错误的步骤。

4. Bob 和 Alice 修正了所有错误并达成了以下的二进制串：0 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1。但是，他们知道 Eve 已经知道了该密钥的某些部分(可能是其中两位)，因此他们选取 $t=2$，使用 4 个位的子集，该子集是从以下随机选取的：

(1,3,10,16) (2,4,11,12) (3,6,8,9) (5,7,14,15) (4,8,13,15) (5,6,7,8)

(2,10,13,15) (1,9,11,14) (6,10,12,14) (4,10,11,12) (5,10,14,16) (3,7,9,11)

那么他们的最终密钥是什么？